

ГОСТ 34.1702.3—92  
(ИСО 8651—3—88)

ГОСУДАРСТВЕННЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ

## МАШИННАЯ ГРАФИКА

СВЯЗЬ ЯДРА ГРАФИЧЕСКОЙ СИСТЕМЫ  
С ЯЗЫКОМ ПРОГРАММИРОВАНИЯ АДА

Издание официальное

МОСКВА  
1993



**к ГОСТ Р 34.1702.3—92 (ИСО 8651-3—88) Информационная технология. Машинная графика. Связь ядра графической системы с языком программирования Ада**

В каком месте	Напечатано	Должно быть
Обложка, титульный лист	ГОСТ 34.1702.3—92	ГОСТ Р 34.1702.3—92

(ИУС № 12 2007 г.)

**ГОСУДАРСТВЕННЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Информационная технология  
**МАШИННАЯ ГРАФИКА**

Связь ядра графической системы  
с языком программирования Ада

Information technology,  
Computer graphics — Graphical  
Kernel System (GKS) language  
bindings — Ada

**ГОСТ Р****34.1702.3 —92****(ИСО 8651—3—88)**

ОКСТУ 0031

Дата введения 01.01.94

Настоящий стандарт устанавливает правила привязки ядра графической системы (ЯГС) (ГОСТ 27817) к языку программирования Ада (ИСО 8651—3) и определяет:

имена и списки параметров процедур на языке Ада, соответствующие функциям ЯГС;

имена типов данных ЯГС в языке Ада;

структуры данных ЯГС в языке Ада;

имена функций обработки ошибок.

Настоящий стандарт не устанавливает:

структуры и методы реализации ЯГС;

требования к операционной среде и оборудованию;

методы связи ЯГС с другими языками программирования, отличными от языка Ада.

**2. ССЫЛКИ**

В качестве описания правил привязки ЯГС к языку программирования Ада использован международный стандарт ИСО 8651—3—88, приведенный в приложении.

---

Издание официальное

★

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен без разрешения Госстандарта России

**МЕЖДУНАРОДНЫЙ СТАНДАРТ**

Системы обработки информации

**МАШИННАЯ ГРАФИКА**

Связь ядра графической системы с языком программирования Ада

ИСО 8651—3—88

**ПРЕДИСЛОВИЕ**

Международный стандарт ИСО 8651—3 разработан Техническим комитетом ИСО/СТК 1 «Системы обработки информации».

ИСО 8651 состоит из следующих частей под общим заголовком «Системы обработки информации. Машинная графика. Связь ядра графической системы (ЯГС) с языками программирования:

Часть 1. Фортран.

Часть 2. Паскаль.

Часть 3. Ада.

**0. ВВЕДЕНИЕ**

Функциональное описание ядра графической системы (ЯГС), содержащееся в ГОСТ 27817 (ИСО 7942), сформулировано независимо от языка программирования способом и должно быть окружено слоем, зависящим от языка программирования (привязка к языку) для использования с определенным языком программирования.

Цель настоящего стандарта части ИСО 8651 — определение стандартной привязки к языку программирования Ада.

**1. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ**

ГОСТ 27817 (ИСО 7942) определяет языконезависимое ЯГС. Для включения его в язык программирования ЯГС помещается на

языковозависимый уровень, отвечающий соответствующим положениям этого языка. Настоящая часть ИСО 8651 определяет такой языковозависимый уровень для языка программирования Ада.

## 2. ССЫЛКИ

ГОСТ 27817 (ИСО 7942—85) Системы обработки информации. Машинная графика. Описание функций ядра графической системы (ЯГС).

ГОСТ 27831 (ИСО 8652—86) Язык программирования Ада.

## 3. СВЯЗЬ ЯГС С ЯЗЫКОМ АДА

Данная связь не предполагает, что компилятор поддерживает любые свойства языка Ада, которые являются зависящими от реализации, но подразумевает, что компилятор будет способен поддерживать декларации, входящие в связывание GKS/Ада. Не делается никаких предположений относительно машинного представления предопределенных типов чисел в Аде.

Данная связь предполагает, что прикладной программист будет применять имя файла ошибок и идентификатор связи, которые будут иметь формат, приемлемый для реализации Ады.

В данной связи не делается предположения относительно формата строки, определяющей имя файла ошибок или идентификатор связи для устройства или метафайла.

### 3.1. Условия соответствия стандарту

Данная связь включает правила, определенные в стандарте ГОСТ 27817 (ИСО 7942) со следующими дополнительными требованиями, специально заданными для реализации ЯГС в языке Ада.

Для определения соответствия или несоответствия реализации данному связыванию установлены следующие критерии:

а) реализация ЯГС в Аде соответствует уровню ЯГС, если она точно выполняет декларации для данного уровня ЯГС и более низких уровней, определенных данным связыванием;

б) семантика реализации должна соответствовать стандарту ЯГС с модификациями и расширениями для Ады, установленными в данном документе;

в) модуль, соответствующий реализуемому уровню ЯГС, должен быть доступен в виде блока библиотеки Ады с именем, определенным в данном документе.

## 3.2. Включение в язык

## 3.2.1. Отображение функций

Все функции ЯГС отображаются в процедуры Ады. Отображение использует «один-в-один» соответствие между функциями ЯГС и процедурами Ады, исключая функции ЯГС «Узнать текущий атрибут примитива» и «Узнать индивидуальный атрибут». Они связаны с отдельной процедурой Ады для каждого из запрашиваемых атрибутов; атрибуты связываются с одной записью.

## 3.2.2. Реализация и зависимость от компьютера

Существует ряд зависимостей от реализаций и компьютера, связанных с компилятором Ады и используемыми процедурами системы. Это будет влиять на переносимость прикладных программ и их использование ЯГС. Прикладному программисту следует придерживаться принятой практики обеспечения переносимости программ на языке Ада, чтобы избежать возникновения проблем при переносе прикладного программного обеспечения на другую систему. К зависимостям от реализаций относятся управление памятью и процессором.

## 3.2.3. Обработка ошибок

Справочные функции используют параметры индикаторов для возврата ошибок и не порождают прерываний. В языке Ада прикладная программа должна гарантировать, что индикаторы ошибок проверяются до попыток обращения к другим параметрам, так как некоторые реализации Ады не порождают прерываний, если обнаружено неопределенное значение.

Требования ЯГС по обработке ошибок можно выразить следующим образом.

1. По умолчанию процедура, названная `ERROR_HANDLING`, будет обеспечивать простую регистрацию ошибок вызовом `ERROR_LOGGING`. Она вызывается из функций ЯГС, обнаруживших ошибку.

2. Процедура `ERROR_HANDLING` может быть заменена пользователем на другую.

Процедура `ERROR_HANDLING` задается, как библиотечная подпрограмма:

```
with GKS_TYPES;
use GKS_TYPES;
procedure ERROR_HANDLING (ERROR_INDICATOR
                           : in ERROR_NUMBER;
                           GKS_FUNCTION : in STRING;
                           ERROR_FILE  : in STRING;
                           : = DEFAULT_ERROR_FILE);
```

Процедура `ERROR_HANDLING` определяется как библиотечная процедура и не декларируется внутри пакета ЯГС.

Данная связь определяет два различных тела для этой программы; каждое должно быть представлено реализацией. Тело по умолчанию — это одна из требуемых ЯГС семантических конструкций. Это просто вызов `ERROR_LOGGING` и возврат. Функция ЯГС должна быть написана так, чтобы она не поддерживала `GKS_ERROR` (это требование реализации). Таким образом, в соответствии с правилами языка Ада прерывание распространяется назад к прикладной программе, вызвавшей функцию `GKS`, в которой обнаружилась ошибка.

Смысл пользовательской замены тела, применяемого по умолчанию, на версию, порождающую прерывание, зависит от менеджера библиотеки Ады. Некоторые реализации поддерживают множество версий тела с одной спецификацией или допускают применение иерархических библиотек с разделением общих блоков. В других реализациях может оказаться необходимым дублировать библиотеку семантических конструкций для каждой версии `ERROR_HANDLING`.

Ошибки `GKS` отображаются в одно прерывание `GKS_ERROR`, декларированное в пакете `GKS`. Предполагаемый стиль в обработке ошибок с использованием прерываний состоит в том, чтобы обеспечить обработчик для прерываний `GKS_ERROR`.

### 3.2.4. Отображение данных

Простые и составные типы данных ЯГС связываются с различными скалярными и составными типами языка Ада. Ограничения на допустимые значения отображаются, где это возможно, в определениях типов. Общее соотношение между типами данных ЯГС и типами данных Ады следующее:

- целые ЯГС отображаются в целые типы Ады;
- действительные ЯГС отображаются в типы с плавающей запятой Ады;
- строки ЯГС отображаются в тип `STRING` Ады или в тип, предназначенный для строк переменной длины;
- указатели ЯГС отображаются в типы записи Ады;
- имена ЯГС отображаются в дискретный тип Ады;
- нумерация ЯГС отображается в тип нумерации Ады;
- вектора ЯГС отображаются в тип записи Ады;
- матрицы (`matrix`) ЯГС отображаются в типы массивов (`array`) Ады;

списки ЯГС из элементов конкретного типа отображаются в приватный тип Ады, декларированный в общем пакете `GKS_LIST_UTILITIES`;

массивы ЯГС отображаются либо в тип неограниченной матрицы, либо в тип записи, предоставляемый для массивов переменной длины;

упорядоченные пары ЯГС отображаются в типы записей Ады; записи данных ЯГС отображаются в приватные типы Ады.

В некоторых случаях набор подпрограмм для обработки записей данных явно определяется данным стандартом. Реализация ЯГС может предоставить другие подпрограммы для манипулирования зависящими от реализации данными.

### 3.2.5. Многозадачность

В определении языка Ада дается явная поддержка параллельности. Модель организации прохождения задачи включает средства для декларирования и выделения задачи и операции, допускающие межадачную связь и синхронизацию.

Стандарт ЯГС не требует и не запрещает в реализации защиту от проблем, которые могут возникнуть при асинхронном доступе к структурам данных ЯГС из параллельных задач. В пользовательской документации по реализации ЯГС должна содержаться информация о том, введена ли защита от таких проблем.

Приложение Д представляет собой пособие для тех, кто хочет, чтобы поддерживались многозадачные прикладные программы. Приложение не является составной частью стандарта, а предоставляет дополнительную информацию.

### 3.2.6. Пакетирование

Стандарт ЯГС определяет девять уровней графической функциональности, где уровень 0a является низшим, а 2c — высшим уровнем. В реализации ЯГС можно сделать систему единой, а можно реализовать каждый уровень отдельно. Для поддержки этой концепции данный стандарт определяет девять пакетов Ады, которые соответствуют каждому уровню ЯГС. Каждый из этих пакетов именуется `package GKS is ... end GKS`, чтобы обеспечить переносимость прикладных программ для уровней ЯГС. Однако содержимое пакетов различается в зависимости от уровня ЯГС, который они реализуют. Каждый из этих пакетов предоставляет подпрограммы, определенные для их уровней и все подпрограммы, заданные в нижних уровнях, как определено в п. 5.1. С каждым из этих пакетов связаны пакеты типов данных, которые обеспечивают декларацию типов для соответствующего уровня, как определено в п. 4.2, а ЯГС определяет прерывания, приведенные в п. 4.3.1. Данные пакеты именуются `package GKS_TYPES is ... end GKS_TYPES`.

Средства библиотек программ Ады следует использовать для обеспечения отделения уровней. Так графические прикладные

программы на Аде, которые используют ЯГС, должны включать соответствующие пакеты ЯГС, которые предоставляют подпрограмму, типы и прерывания для данного уровня компилирования и включением соответствующей библиотеки Ады, которая содержит данный уровень ЯГС. Например, прикладная программа, которая использует уровень 0a, могла бы выглядеть следующим образом:

```
with GKS;
use GKS_TYPES;
procedure APPLICATION is
begin
null;
end APPLICATION.
```

Далее программа компилируется и связывается с библиотекой программ Ады, которая соответствует уровню 0a.

Данный стандарт определяет и ряд дополнительных блоков Ады. Ими являются общие пакеты:

```
GKS_COORDINATE_SYSTEM;
GKS_LIST_UTILITIES.
```

Эти общие пакеты поддерживают типы деклараций в пакете GKS\_TYPE, описанном выше. GKS\_COORDINATE\_SYSTEM представляет собой общий пакет, в котором определен ассортимент типов для поддержания каждой координатной системы ЯГС. GKS\_LIST\_UTILITIES является также общим пакетом, который предоставляет декларации списков и операции для типов списков, которые соответствуют типам списков ЯГС.

### 3.2.7. Среда прикладных программ

Прикладная программа, применяющая реализацию ЯГС в языке Ада, должна будет знать среду, в которой находятся и ЯГС и прикладная программа.

Одним таким интерфейсом является библиотека программ Ады. Язык Ада требует, чтобы прикладная программа имела доступ к библиотеке программ, в которой размещается программное обеспечение ЯГС. Справочное руководство по языку Ада ИСО 8652 не определяет, должна быть одна библиотека или их может быть много, как предоставляется доступ к библиотеке и так далее. Пользовательская документация по реализации ЯГС должна содержать информацию о том, где в системе находится библиотека ЯГС и как осуществляется к ней доступ.

Интерфейсы ввода/вывода также являются зависящими от реализации и требуют описания в пользовательской документации. Должны быть включены в документацию интерфейсы с фай-

ловой системой. Особенно это относится к интерфейсу с файлом ошибок ЯГС, а также к памяти метафайла.

### 3.2.8. Регистрация\*

Стандарт ЯГС резервирует различные области величин для регистрации в качестве графических элементов. Зарегистрированные графические элементы будут встроены в Аду (и другие языки программирования). Зарегистрированные встроенные элементы будут согласовываться со связыванием, представленным в данном документе.

## 4. ТАБЛИЦЫ

### 4.1. Процедуры

Таблица 1

Сокращения, используемые в именах процедур

ASF	Флаг выборки атрибутов
CHAR	Литера
ESC	Расширение (escape)
GDP	Обобщенный примитив вывода
GKS	Ядро графической системы
GKSM	Метафайл ядра графической системы
ID	Идентификатор
INQ	Справочная функция
MAX	Максимум
UGDP	Незарегистрированный обобщенный примитив вывода
UESC	Незарегистрированное расширение
WS	Станция

Таблица 2

Список процедур, использующих сокращения

ASF	INQ_LIST_OF_ASF SET_ASF
CHAR	INQ_CHAR_BASE_VECTOR INQ_CHAR_EXPANSION_FACTOR INQ_CHAR_HEIGHT INQ_CHAR_WIDTH INQ_CHAR_SPACING INQ_CHAR_UP_VECTOR

\* В соответствии с правилами назначения и работы Органов регистрации в директивах ИСО Совет ИСО для этой части (ИСО 8651) назначил Национальное бюро стандартов (Научно-производственный институт ЭВМ) как орган регистрации A266 Technology Building, Gaithersburg, MD, 20899, USA.

	SET_CHAR_EXPANSION_FACTOR
	SET_CHAR_HEIGHT
	SET_CHAR_SPACING
	SET_CHAR_UP_VECTOR
ESC	ESC
	UESC
GDP	GDP
	INQ_GDP
	INQ_LIST_OF_AVAILABLE_GDP
	UGDP
GKS	CLOSE_GKS
	EMERGENCY_CLOSE_GKS
	INQ_LEVEL_OF_GKS
	OPEN_GKS
GKSM	GET_ITEM_TYPE_FROM_GKSM
	READ_ITEM_FROM_GKSM
	WRITE_ITEM_TO_GKSM
ID	INQ_CURRENT_PICK_ID_VALUE
	SET_PICK_ID
IND	INQ_CHAR_BASE_VECTOR
	INQ_CHAR_EXPANSION_FACTOR
	INQ_CHAR_HEIGHT
	INQ_CHAR_WIDTH
	INQ_CHAR_SPACING
	INQ_CHAR_UP_VECTOR
	INQ_CHOICE_DEVICE_STATE
	INQ_CLIPPING
	INQ_COLOUR_FACILITES
	INQ_COLOUR_REPRESENTATION
	INQ_CURRENT_NORMALIZATION_TRANSFOR-
	MATION_NUMBER
	INQ_CURRENT_INDIVIDUAL_ATTRIBUTE_VALUES
	INQ_CURRENT_PICK_ID_VALUE
	INQ_CURRENT_PRIMITIVE_ATTRIBUTE_VALUES
	INQ_DEFAULT_CHOICE_DEVICE_DATA
	INQ_DEFAULT_DEFERRAL_STATE_VALUES
	INQ_DEFAULT_LOCATOR_DEVICE_DATA
	INQ_DEFAULT_PICK_DEVICE_DATA
	INQ_DEFAULT_STRING_DEVICE_DATA
	INQ_DEFAULT_STROKE_DEVICE_DATA
	INQ_DEFAULT_VALUATOR_DEVICE_DATA
	INQ_DISPLAY_SPACE_SIZE
	INQ_DYNAMIC_MODIFICATION_OF_SEGMENT-
	_ATTRIBUTES
	INQ_DYNAMIC_MODIFICATION_OF_WS-
	_ATTRIBUTES

INQ\_FILL\_AREA\_COLOUR INDEX  
INQ\_FILL\_AREA\_FACILITIES  
INQ\_FILL\_AREA\_INDEX  
INQ\_FILL\_AREA\_INTERIOR\_STYLE  
INQ\_FILL\_AREA\_REPRESENTATION  
INQ\_FILL\_AREA\_STYLE\_INDEX  
INQ\_GDP  
INQ\_INPUT\_QUEUE\_OVERFLOW  
INQ\_LEVEL\_OF\_GKS  
INQ\_LIST\_OF\_ASF  
INQ\_LINETYPE  
INQ\_LINEWIDTH\_SCALE\_FACTOR  
INQ\_LIST\_OF\_AVAILABLE\_GDP  
INQ\_LIST\_OF\_AVAILABLE\_WS\_TYPE  
INQ\_LIST\_OF\_COLOUR\_INDICES  
INQ\_LIST\_OF\_FILL\_ARFA\_INDICES  
INQ\_LIST\_OF\_NORMALIZATION\_TRANSFOR-  
MATION\_NUMBER  
INQ\_LIST\_OF\_PATTERN\_INDICES  
INQ\_LIST\_OF\_POLYLINE\_INDICES  
INQ\_LIST\_OF\_POLYMARKER\_INDICES  
INQ\_LIST\_OF\_TEXT\_INDICES  
INQ\_LOCATOR\_DEVICE\_STATE  
INQ\_MAX\_LENGTH\_OF\_WS\_STATE\_TABLES  
INQ\_MAX\_NORMALIZATION\_TRANSFORMATION\_  
NUMBER  
INQ\_MORE\_SIMULTANEOUS\_EVENTS  
INQ\_NAME\_OF\_OPEN\_SEGMENT  
INQ\_NORMALIZATION\_TRANSFORMATION  
INQ\_NUMBER\_OF\_SEGMENT\_PRIORITIFS\_  
SUPPORTED  
INQ\_NUMBER\_OF\_AVAILABLE\_LOGICAL\_INPUT\_  
DEVICES  
INQ\_OPERATING\_STATE\_VALUE  
INQ\_PATTERN\_FACILITIES  
INQ\_PATTERN\_HEIGHT\_VECTOR  
INQ\_PATTERN\_REFERENCE\_POINT  
INQ\_PATTERN\_REPRESENTATION  
INQ\_PATTERN\_WIDTH\_VECTOR  
INQ\_PICK\_DEVICE\_STATE  
INQ\_PIXEL  
INQ\_PIXEL\_ARRAY  
INQ\_PIXEL\_ARRAY\_DIMENSIONS  
INQ\_POLYLINE\_COLOUR\_INDEX  
INQ\_POLYLINE\_FACILITIES  
INQ\_POLYLINE\_INDEX  
INQ\_POLYLINE\_REPRESENTATION  
INQ\_POLYMARKER\_REPRESENTATION  
INQ\_POLYMARKER\_COLOUR\_INDEX  
INQ\_POLYMARKER\_INDEX  
INQ\_POLYMARKER\_FACILITIES  
INQ\_POLYMARKER\_SIZE\_SCALE\_FACTOR

	INQ_POLYMARKER_TYPE
	INQ_PREDEFINED_COLOUR_REPRESENTATION
	INQ_PREDEFINED_FILL_AREA_REPRESENTATION
	INQ_PREDEFINED_PATTERN_REPRESENTATION
	INQ_PREDEFINED_POLYLINE_REPRESENTATION
	INQ_PREDEFINED_POLYMARKER_REPRESENTATION
	INQ_PREDEFINED_TEXT_REPRESENTATION
	INQ_SEGMENT_ATTRIBUTES
	INQ_SET_OF_ACTIVE_WS
	INQ_SET_OF_ASSOCIATED_WS
	INQ_SET_OF_OPEN_WS
	INQ_SET_OF_SEGMENT_NAMES_IN_USE
	INQ_SET_OF_SEGMENT_NAMES_ON_WS
	INQ_STRING_DEVICE_STATE
	INQ_STROKE_DEVICE_STATE
	INQ_TEXT_ALIGNMENT
	INQ_TEXT_COLOUR_INDEX
	INQ_TEXT_EXTENT
	INQ_TEXT_FACILITIES
	INQ_TEXT_FONT_AND_PRECISION
	INQ_TEXT_INDEX
	INQ_TEXT_PATH
	INQ_TEXT_REPRESENTATION
	INQ_VALUATOR_DEVICE_STATE
	INQ_WS_CATEGORY
	INQ_WS_CLASSIFICATION
	INQ_WS_CONNECTION_AND_TYPE
	INQ_WS_DEFERRAL_AND_UPDATE_STATES
	INQ_WS_MAX_NUMBER
	INQ_WS_STATE
	INQ_WS_TRANSFORMATION
MAX	INQ_MAX_LENGTH_OF_WS_STATE_TABLES
	INQ_MAX_NORMALIZATION_TRANSFORMATION_NUMBER
	INQ_WS_MAX_NUMBERS
WS	ACTIVATE_WS
	ASSOCIATE_SEGMENT_WITH_WS
	CLEAR_WS
	GLOSE_WS
	COPY_SEGMENT_TO_WS
	DEACTIVATE_WS
	DELETE_SEGMENT_FROM_WS
	INQ_DYNAMIC_MODIFICATION_OF_WS_ATTRIBUTES
	INQ_LIST_OF_AVAILABLE_WS_TYPE
	INQ_MAX_LENGTH_OF_WS_STATE_TABLES
	INQ_SET_OF_ACTIVE_WS
	INQ_SET_OF_ASSOCIATED_WS

INQ\_SET\_OF\_OPEN\_WS  
 INQ\_SET\_OF\_SEGMENT\_NAMES\_ON\_WS  
 INQ\_WS\_CATEGORY  
 INQ\_WS\_CLASSIFICATION  
 INQ\_WS\_CONNECTION\_AND\_TYPE  
 INQ\_WS\_DEFERRAL\_AND\_UPDATE\_STATES  
 INQ\_WS\_MAX\_NUMBER  
 INQ\_WS\_STATE  
 INQ\_WS\_TRANSFORMATION  
 OPEN\_WS  
 REDRAW\_ALL\_SEGMENTS\_ON\_WS  
 SET\_WS\_VIEWPORT  
 SET\_WS\_WINDOW  
 UPDATE\_WS

Таблица 3

Функции ЯГС и имена соответствующих процедур в Аде

Имя в Аде	Функция ЯГС
ACCUMULATE_TRANSFORMATION-MATRIX	ВЫЧИСЛИТЬ РЕЗУЛЬТИРУЮЩУЮ МАТРИЦУ ПРЕОБРАЗОВАНИЙ
ACTIVATE_WS	АКТИВИРОВАТЬ СТАНЦИЮ
ASSOCIATE_SEGMENT_WITH_WS	СВЯЗАТЬ СЕГМЕНТ СО СТАНЦИЕЙ
AWAIT_EVENT	ОЖИДАТЬ СОБЫТИЕ
CELL_ARRAY	МАТРИЦА ЯЧЕЕК
CLEAR_WS	ОЧИСТИТЬ ИЗОБРАЖЕНИЕ НА СТАНЦИИ
CLOSE_GKS	ЗАКРЫТЬ ЯГС
CLOSE_SEGMENT	ЗАКРЫТЬ СЕГМЕНТ
CLOSE_WS	ЗАКРЫТЬ СТАНЦИЮ
COPY_SEGMENT_TO_WS	ВЫВЕСТИ КОПИЮ СЕГМЕНТА НА СТАНЦИЮ
CREATE_SEGMENT	СОЗДАТЬ СЕГМЕНТ
DEACTIVATE_WS	ДЕАКТИВИРОВАТЬ СТАНЦИЮ
DELETE_SEGMENT	УНИЧТОЖИТЬ СЕГМЕНТ
DELETE_SEGMENT_FROM_WS	УДАЛИТЬ СЕГМЕНТ СО СТАНЦИИ
EMERGENCY_CLOSE_GKS	АВАРИЙНО ЗАКРЫТЬ ЯГС
ERROR_HANDLING	ОБРАБОТАТЬ ОШИБКУ
ERROR_LOGGING	ЗАРЕГИСТРИРОВАТЬ ОШИБКУ
ESCAPE	РАСШИРЕНИЕ
EVALUATE_TRANSFORMATION_MATRIX	СФОРМИРОВАТЬ МАТРИЦУ ПРЕОБРАЗОВАНИЙ
FILL_AREA	ПОЛИГОНАЛЬНАЯ ОБЛАСТЬ
FLUSH_DEVICE_EVENTS	УДАЛИТЬ СОБЫТИЯ ОТ УСТРОЙСТВА
GDP	ОБОБЩЕННЫЙ ПРИМИТИВ ВЫВОДА (ОПВ)
GET_CHOICE	ПОЛУЧИТЬ АЛЬТЕРНАТИВУ
GET_ITEM_TYPE_FROM_GKSM	ПОЛУЧИТЬ ТИП ЗАПИСИ ИЗ ЯГС
GET_LOCATOR	ПОЛУЧИТЬ ПОЗИЦИЮ
GET_PICK	ПОЛУЧИТЬ УКАЗАТЕЛЬ

Продолжение табл. 3

Имя в Аде	Функция ЯГС
GET_STRING GET_STROKE	ПОЛУЧИТЬ СТРОКУ ПОЛУЧИТЬ ПОСЛЕДОВАТЕЛЬ- НОСТЬ ПОЗИЦИИ
GET_VALUATOR INITIALISE_CHOICE	ПОЛУЧИТЬ ЧИСЛО ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТ- ВО ВЫБОРА
INITIALISE_LOCATOR	ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТ- ВО ВВОДА ПОЗИЦИИ
INITIALISE_PICK	ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТ- ВО УКАЗАНИЯ
INITIALISE_STRING	ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТ- ВО ВВОДА СТРОКИ
INITIALISE_STROKE	ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТ- ВО ВВОДА ПОСЛЕДОВАТЕЛЬНОС- ТИ ПОЗИЦИИ
INITIALISE_VALUATOR	ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТ- ВО ВВОДА ЧИСЛА
INQ_CHOICE_DEVICE_STATE	УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА ВЫБОРА
INQ_CLIPPING INQ_COLOUR_FACILITIES	УЗНАТЬ ЗНАЧЕНИЯ ОТСЕЧЕНИЯ УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДС- ТАВЛЕНИЯ ЦВЕТА
INQ_COLOR_REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ЦВЕТА
INQ_CURRENT_INDIVIDUAL_ _ATTRIBUTE_VALUES	УЗНАТЬ ТЕКУЩИЕ ЗНАЧЕНИЯ ИН- ДИВИДУАЛЬНЫХ АТТРИБУТОВ

Функция ЯГС «Узнать значение текущего индивидуального атрибута» отображается в следующие функции:

INQ\_CHAR\_EXPANSION\_FACTOR  
 INQ\_CHAR\_SPACING  
 INQ\_FILL\_AREA\_COLOUR\_INDEX  
 INQ\_FILL\_AREA\_INTERIOR\_STYLE  
 INQ\_LINETYPE  
 INQ\_LINEWIDTH\_SCALE\_FACTOR  
 INQ\_LIST\_OF\_ASF  
 INQ\_POLYLINE\_COLOUR\_INDEX  
 INQ\_POLYMARKER\_COLOUR\_INDEX  
 INQ\_POLYMARKER\_SIZE\_SCALE\_FACTOR  
 INQ\_POLYMARKER\_TYPE  
 INQ\_TEXT\_COLOUR\_INDEX  
 INQ\_TEXT\_FONT\_AND\_PRECISION

INQ_CURRENT_NORMALIZATION_TRANSFORMATION_NUMBER	УЗНАТЬ НОМЕР ТЕКУЩЕГО ПРЕОБРАЗОВАНИЯ НОРМИРОВАНИЯ
INQ_CURRENT_PICK_ID_VALUE	УЗНАТЬ ЗНАЧЕНИЕ ИДЕНТИФИКАТОРА УКАЗАНИЯ
INQ_CURRENT_PRIMITIVE_ATTRIBUTE_VALUES	УЗНАТЬ ТЕКУЩИЕ ЗНАЧЕНИЯ АТТРИБУТОВ ПРИМИТИВОВ

Функция ЯГС «Узнать текущие значения атрибутов примитивов» отображается в следующие функции:

INQ\_CHAR\_BASE\_VECTOR  
 INQ\_CHAR\_HEIGHT  
 INQ\_CHAR\_WIDTH  
 INQ\_CHAR\_UP\_VECTOR  
 INQ\_FILL\_AREA\_INDEX  
 INQ\_PATTERN\_HEIGHT\_VECTOR  
 INQ\_PATTERN\_REFERENCE\_POINT  
 INQ\_PATTERN\_WIDTH\_VECTOR  
 INQ\_POLYLINE\_INDEX  
 INQ\_POLYMARKER\_INDEX  
 INQ\_TEXT\_ALIGNMENT  
 INQ\_TEXT\_INDEX  
 INQ\_TEXT\_PATH

*Продолжение табл. 3*

Имя в Ада	Функция ЯГС
INQ_DEFAULT_CHOICE_DEVICE_DATA	УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ УСТРОЙСТВА ВЫБОРА
INQ_DEFAULT_DEFERRAL_STATE_VALUES	УЗНАТЬ РЕЖИМ ЗАДЕРЖКИ ПО УМОЛЧАНИЮ
INQ_DEFAULT_LOCATOR_DEVICE_DATA	УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ УСТРОЙСТВА ВВОДА ПОЗИЦИИ
INQ_DEFAULT_PICK_DEVICE_DATA	УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ УСТРОЙСТВА УКАЗАНИЯ
INQ_DEFAULT_STRING_DEVICE_DATA	УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ УСТРОЙСТВА ВВОДА СТРОКИ
INQ_DEFAULT_STROKE_DEVICE_DATA	УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ УСТРОЙСТВА ВВОДА ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИИ

Имя в ЛЭФ	Функция ЯГС
INQ_DEFAULT_VALUATOR_ _DEVICE_DATA	УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ УСТРОЙСТВА ВВО- ДА ЧИСЛА
INQ_DISPLAY_SPACE_SIZE	УЗНАТЬ РАЗМЕР НОСИТЕЛЯ ИЗОБ- РАЖЕНИЯ
INQ_DYNAMIC_MODIFICATION_ _OF_SEGMENT_ATTRIBUTES	УЗНАТЬ СПОСОБ ДИНАМИЧЕСКОЙ МОДИФИКАЦИИ АТТРИБУТОВ СЕГ- МЕНТОВ
INQ_DYNAMIC_MODIFICATION_ _OF_WS_ATTRIBUTES	УЗНАТЬ СПОСОБ ДИНАМИЧЕСКО- ГО ОБНОВЛЕНИЯ ХАРАКТЕРИС- ТИК ИЗОБРАЖЕНИЯ НА СТАН- ЦИИ
INQ_FILL_AREA_FACILITIES	УЗНАТЬ ВОЗМОЖНОСТИ ПРЕД- СТАВЛЕНИЯ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ
INQ_FILL_AREA_ _REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИ- ГОНАЛЬНОЙ ОБЛАСТИ
INQ_GDP	УЗНАТЬ ХАРАКТЕРИСТИКИ ОБОБ- ЩЕННОГО ПРИМИТИВА ВЫВОДА
INQ_INPUT_QUEUE_OVERFLOW	УЗНАТЬ НАЛИЧИЕ ПЕРЕПОЛНЕНИЯ ОЧЕРЕДИ СОБЫТИЙ
INQ_LEVEL_OF_GKS	УЗНАТЬ УРОВЕНЬ ЯГС
INQ_LIST_OF_AVAILABLE_GDP	УЗНАТЬ ИДЕНТИФИКАТОРЫ ДОС- ТУПНЫХ ОБОБЩЕННЫХ ПРИМИ- ТИВОВ ВЫВОДА
INQ_LIST_OF_AVAILABLE_WS_ _TYPES	УЗНАТЬ ДОСТУПНЫЕ ТИПЫ СТАН- ЦИИ
INQ_LIST_OF_COLOUR_INDICES	УЗНАТЬ ИНДЕКС ЦВЕТА
INQ_LIST_OF_FILL_AREA_ _INDICES	УЗНАТЬ ИНДЕКСЫ ПОЛИГОНАЛЬ- НОЙ ОБЛАСТИ
INQ_LIST_OF_NORMALIZATION_ _TRANSFORMATION_NUMBERS	УЗНАТЬ СПИСОК ПРЕОБРАЗОВА- НИЯ НОРМИРОВАНИЯ
INQ_LIST_OF_PATTERN_ _INDICES	УЗНАТЬ ИНДЕКСЫ ШАБЛОНА
INQ_LIST_OF_POLYLINE_ _INDICES	УЗНАТЬ ИНДЕКСЫ ЛОМАНОЙ
INQ_LIST_OF_POLYMARKER_ _INDICES	УЗНАТЬ ИНДЕКСЫ ПОЛИМАРКЕРА
INQ_LIST_OF_TEXT_INDICES	УЗНАТЬ ИНДЕКСЫ ТЕКСТА
INQ_LOCATOR_DEVICE_STATE	УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА ВВОДА ПОЗИЦИИ
INQ_MAX_LENGTH_OF_WS_ _STATE_TABLES	УЗНАТЬ ДЛИНУ ТАБЛИЦ ХАРАК- ТЕРИЗУЮЩИХ СТАНЦИЮ
INQ_MAX_NORMALIZATION_ _TRANSFORMATION_NUMBER	УЗНАТЬ МАКСИМАЛЬНЫЙ НОМЕР ПРЕОБРАЗОВАНИЯ НОРМИРОВА- НИЯ
INQ_MORE_SIMULTANEOUS_ _EVENTS	УЗНАТЬ НАЛИЧИЕ ОДНОВРЕМЕН- НЫХ СОБЫТИЙ

Имя в Адр	Функция ЯТС
INQ_NAME_OF_OPEN_SEGMENT	УЗНАТЬ ИМЯ ОТКРЫТОГО СЕГМЕНТА
INQ_NORMATIZATION_TRANSFORMATION	УЗНАТЬ ПРЕОБРАЗОВАНИЯ НОРМИРОВАНИЯ
INQ_NUMBER_OF_AVAILABLE_LOGICAL_INPUT_DEVICES	УЗНАТЬ ЧИСЛО ДОПУСТИМЫХ УСТРОЙСТВ ВВОДА
INQ_NUMBER_OF_SEGMENT_PRIORITIES_SUPPORTED	УЗНАТЬ ДОПУСТИМОЕ ЧИСЛО ПРИОРИТЕТОВ СЕГМЕНТОВ
INQ_OPERATING STATE VALUE	УЗНАТЬ ФУНКЦИОНАЛЬНОЕ СОСТОЯНИЕ
INQ_PATTERN_FACILITIES	УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ ШАБЛОНА
INQ_PATTERN_REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ШАБЛОНА
INQ_PICK_DEVICE_STATE	УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА УКАЗАНИЯ
INQ_PIXEL	УЗНАТЬ ЦВЕТ ПИКСЕЛЯ
INQ_PIXEL_ARRAY	УЗНАТЬ МАТРИЦУ ПИКСЕЛЕН
INQ_PIXEL_ARRAY_DIMENSION	УЗНАТЬ РАЗМЕРНОСТЬ МАТРИЦЫ ПИКСЕЛЕЙ
INQ_POLYLINE_FACILITIES	УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ ЛОМАННОЙ
INQ_POLYLINE REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ЛОМАННОЙ
INQ_POLYMARKER_FACILITIES	УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ ПОЛИМАРКЕРА
INQ_POLYMARKER_REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИМАРКЕРА
INQ_PREDEFINED_COLOUR_REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ЦВЕТА ПО УМОЛЧАНИЮ
INQ_PREDEFINED_FILL_AREA_REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ ПО УМОЛЧАНИЮ
INQ_PREFERRED_PATTERN_REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ШАБЛОНА ПО УМОЛЧАНИЮ
INQ_PREDEFINED_POLYLINE_REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ЛОМАННОЙ ПО УМОЛЧАНИЮ
INQ_PREDEFINED_POLYMARKER_REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИМАРКЕРА ПО УМОЛЧАНИЮ
INQ_PREDEFINED_TEXT_REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ТЕКСТА ПО УМОЛЧАНИЮ
INQ_SEGMENT_ATTRIBUTES	УЗНАТЬ АТТРИБУТЫ СЕГМЕНТА
INQ_SET_OF_ACTIVE_WS	УЗНАТЬ НАБОР АКТИВНЫХ СТАНЦИЙ
INQ_SET_OF_ASSOCIATED_WS	УЗНАТЬ СТАНЦИИ СВЯЗАННЫЕ С СЕГМЕНТОМ
INQ_SET_OF_OPEN_WS	УЗНАТЬ НАБОР ОТКРЫТЫХ СТАНЦИЙ

Имя в Лдо	Функция ЯГС
INQ_SET_OF_SEGMENT_NAMES_ _IN_USE	УЗНАТЬ ИМЕНА СУЩЕСТВУЮЩИХ СЕГМЕНТОВ
INQ_SET_SEGMENT_NAMES_ON_ _WS	УЗНАТЬ ИМЕНА СЕГМЕНТОВ, ХРАНИМЫХ НА СТАНЦИИ
INQ_STRING_DEVICE_STATE	УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА ВВОДА СТРОКИ
INQ_STROKE_DEVICE_STATE	УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА ВВОДА ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИИ
INQ_TEXT_EXTENT	УЗНАТЬ ГАБАРИТЫ ТЕКСТА
INQ_TEXT_FACILITIES	УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ ТЕКСТА
INQ_TEXT_REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ТЕКСТА
INQ_VALUATOR_DEVICE_STATE	УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА ВВОДА ЧИСЛА
INQ_WS_CATEGORY	УЗНАТЬ КАТЕГОРИЮ СТАНЦИИ
INQ_WS_CLASSIFICATION	УЗНАТЬ КЛАСС СТАНЦИИ
INQ_WS_CONNECTION_AND_ _TYPE	УЗНАТЬ ТИП И ИДЕНТИФИКАТОР СВЯЗИ СТАНЦИИ
INQ_WS_DEFERRAL_AND_ _UPDATE_STATES	УЗНАТЬ РЕЖИМЫ ЗАДЕРЖКИ И ОБНОВЛЕНИЯ СТАНЦИИ
INQ_WS_MAX_NUMBERS	УЗНАТЬ ДОПУСТИМОЕ ЧИСЛО СТАНЦИИ
INQ_WS_STATE	УЗНАТЬ СОСТОЯНИЕ СТАНЦИИ
INQ_WS_TRANSFORMATION	УЗНАТЬ ПРЕОБРАЗОВАНИЕ СТАНЦИИ
INSERT_SEGMENT	ВСТАВИТЬ СЕГМЕНТ
INTERPRET_ITEM	ИНТЕРПРЕТИРОВАТЬ ЗАПИСЬ
MESSAGE	СООБЩЕНИЕ
OPEN_GKS	ОТКРЫТЬ ЯГС
OPEN_WS	ОТКРЫТЬ СТАНЦИЮ
POLYLINE	ЛОМАНАЯ
POLYMARKER	ПОЛИМАРКЕР
READ_ITEM_FROM_GKSM	ПРОЧИТАТЬ ЗАПИСЬ ИЗ ЯГС
REDRAW_ALL_SEGMENTS_ON_ _WS	ПЕРЕРИСОВАТЬ ВСЕ СЕГМЕНТЫ НА СТАНЦИИ
RENAME_SEGMENT	ПЕРЕИМЕНОВАТЬ СЕГМЕНТ
REQUEST_CHOICE	ЗАПРОСИТЬ УСТРОЙСТВО ВЫБОРА
REQUEST_LOCATOR	ЗАПРОСИТЬ ВВОД ПОЗИЦИИ
REQUEST_PICK	ЗАПРОСИТЬ УКАЗАНИЯ
REQUEST_STRING	ЗАПРОСИТЬ ВВОД СТРОКИ
REQUEST_STROKE	ЗАПРОСИТЬ ВВОД ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИИ
REQUEST_VALUATOR	ЗАПРОСИТЬ ВВОД ЧИСЛА
SAMPLE_CHOICE	ОПРОСИТЬ ВЫБОР
SAMPLE_LOCATOR	ОПРОСИТЬ ВВОД ПОЗИЦИИ
SAMPLE_PICK	ОПРОСИТЬ УКАЗАНИЕ
SAMPLE_STRING	ОПРОСИТЬ ВВОД СТРОКИ

Имя в Адр	Функция ЯЭС
SAMPLE_STROKE	ОПРОСИТЬ ВВОД ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИИ
SAMPLE_VALUATOR	ОПРОСИТЬ ВВОД ЧИСЛА
SELECT_NORMALIZATION_TRANSFORMATION	ВЫБРАТЬ ПРЕОБРАЗОВАНИЕ НОРМИРОВАНИЯ
SET_ASF	ЗАДАТЬ ФЛАГИ ВЫБОРКИ АТРИБУТОВ
SET_CHAR_EXPANSION_FACTOR	ЗАДАТЬ МАСШТАБ РАСШИРЕНИЯ ЛИТЕРЫ
SET_CHAR_HEIGHT	ЗАДАТЬ ВЫСОТУ ЛИТЕР
SET_CHAR_SPACING	ЗАДАТЬ МЕЖЛИТЕРНЫЙ ПРОСВЕТ
SET_CHAR_UP_VECTOR	ЗАДАТЬ ВЕРТИКАЛЬ ЛИТЕРЫ
SET_CHOICE_MODE	ЗАДАТЬ РЕЖИМ УСТРОЙСТВА ВЫБОРА
SET_CLIPPING_INDICATOR	ЗАДАТЬ ИНДИКАТОР ОТСЕЧЕНИЯ
SET_COLOUR_REPRESENTATION	ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ЦВЕТА
SET_DEFERRAL_STATE	ЗАДАТЬ РЕЖИМ ЗАДЕРЖКИ
SET_DETECTABILITY	ЗАДАТЬ ЧУВСТВИТЕЛЬНОСТЬ К УКАЗАНИЮ
SET_FILL_AREA_COLOUR_INDEX	ЗАДАТЬ ИНДЕКС ЦВЕТА ПОЛИГОНАЛЬНОЙ ОБЛАСТИ
SET_FILL_AREA_INDEX	ЗАДАТЬ ИНДЕКС ПОЛИГОНАЛЬНОЙ ОБЛАСТИ
SET_FILL_AREA_INTERIOR_STYLE	ЗАДАТЬ ВИД ЗАПОЛНЕНИЯ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ
SET_FILL_AREA_REPRESENTATION	ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ
SET_FILL_AREA_STYLE_INDEX	ЗАДАТЬ ИНДЕКС ЗАПОЛНИТЕЛЯ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ
SET_HIGHLIGHTING	ЗАДАТЬ ВЫДЕЛЕНИЕ
SET_LINETYPE	ЗАДАТЬ ТИП ЛИНИИ
SET_LINewidth_SCALE_FACTOR	ЗАДАТЬ МАСШТАБ ТОЛЩИНЫ ЛИНИИ
SET_LOCATOR_MODE	ЗАДАТЬ РЕЖИМ УСТРОЙСТВА ВВОДА ПОЗИЦИИ
SET_MARKER_SIZE_SCALE_FACTOR	ЗАДАТЬ МАСШТАБ МАРКЕРА
SET_MARKER_TYPE	ЗАДАТЬ ТИП МАРКЕРА
SET_PATTERN_REFERENCE_POINT	ЗАДАТЬ ТОЧКУ ПРИВЯЗКИ ШАБЛОНА
SET_PATTERN REPRESENTATION	ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ШАБЛОНА
SET_PATTERN_SIZE	ЗАДАТЬ РАЗМЕР ШАБЛОНА
SET_PICK_ID	ЗАДАТЬ ИДЕНТИФИКАТОР УКАЗАНИЯ
SET_PICK_MODE	ЗАДАТЬ РЕЖИМ УСТРОЙСТВА УКАЗАНИЯ
SET_POLYLINE_COLOUR_INDEX	ЗАДАТЬ ИНДЕКС ЦВЕТА ЛОМАННОЙ

Имя в Аде	Функция ЯГС
SET_POLYLINE_INDEX	ЗАДАТЬ ИНДЕКС ЛОМАНОЙ
SET_POLYLINE REPRESENTATION	ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ЛОМАНОЙ
SET_POLYMARKER_COLOUR_INDEX	ЗАДАТЬ ИНДЕКС ЦВЕТА ПОЛИМАРКЕРА
SET_POLYMARKER_INDEX	ЗАДАТЬ ИНДЕКС ПОЛИМАРКЕРА
SET_POLYMARKER REPRESENTATION	ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИМАРКЕРА
SET_SEGMENT_PRIORITY	ЗАДАТЬ ПРИОРИТЕТ СЕГМЕНТА
SET_SEGMENT_TRANSFORMATION	ЗАДАТЬ ПРЕОБРАЗОВАНИЕ СЕГМЕНТА
SET_STRING_MODE	ЗАДАТЬ РЕЖИМ УСТРОЙСТВА ВВОДА СТРОКИ
SET_STROKE_MODE	ЗАДАТЬ РЕЖИМ УСТРОЙСТВА ВВОДА ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИЙ
SET_TEXT_ALIGNMENT	ЗАДАТЬ ВЫРАВНИВАНИЕ ТЕКСТА
SET_TEXT_COLOUR_INDEX	ЗАДАТЬ ИНДЕКС ЦВЕТА ТЕКСТА
SET_TEXT_FONT_AND_PRECISION	ЗАДАТЬ ШРИФТ И ТОЧНОСТЬ ТЕКСТА
SET_TEXT_INDEX	ЗАДАТЬ ИНДЕКС ТЕКСТА
SET_TEXT_PATH	ЗАДАТЬ НАПРАВЛЕНИЕ ТЕКСТА
SET_TEXT_REPRESENTATION	ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ТЕКСТА
SET_VALUATOR_MODE	ЗАДАТЬ РЕЖИМ УСТРОЙСТВА ВВОДА ЧИСЛА
SET_VIEWPORT	ЗАДАТЬ ПОЛЕ ВЫВОДА
SET_VIEWPORT_INPUT_PRIORITY	ЗАДАТЬ ПРИОРИТЕТ ПОЛЯ ВЫВОДА ПРИ ВВОДЕ
SET_VISIBILITY	ЗАДАТЬ ВИДИМОСТЬ
SET_WINDOW	ЗАДАТЬ ОКНО
SET_WS_VIEWPORT	ЗАДАТЬ ПОЛЕ ВЫВОДА СТАНЦИИ
SET_WS_WINDOW	ЗАДАТЬ ОКНО СТАНЦИИ
TEXT	ТЕКСТ
UPDATE WS	ОБНОВИТЬ ИЗОБРАЖЕНИЕ НА СТАНЦИИ
WRITE_ITEM_TO_GKSM	ЗАПИСЬ В МЕТАФАЙЛ

*Функции ЯГС, упорядоченные по алфавиту*

Функции находятся в том же порядке, в котором перечислены имена процедур, соответствующие именам функций ЯГС. В табл. 3 перечислены по алфавиту имена функций ЯГС.

## Список функций ЯГС по уровням и по алфавиту

## Уровень 0а

ACTIVATE\_WS  
 CELL\_ARREY  
 CLEAR\_WS  
 CLOSE\_GKS  
 CLOSE\_WS  
 DEACTIVATE\_WS  
 EMERGENCY\_CLOSE\_GKS  
 ERROR\_HANDLING  
 ERROR\_LOGGING  
 ESCAPE  
 FILL\_AREA  
 GDP  
 GET\_ITEM\_TYPE\_FROM\_GKSM  
 INQ\_CLIPPING  
 INQ\_COLOUR\_FACILITIES  
 INQ\_COLOUR\_REPRESENTATION  
 INQ\_CURRENT\_INDIVIDUAL\_ATTRIBUTE\_VALUES

Функция ЯГС «Узнать значение текущего индивидуального атрибута» отображается в следующие функции:

INQ\_CHAR\_EXPANSION\_FACTOR  
 INQ\_CHAR\_SPACING  
 INQ\_FILL\_AREA\_COLOUR\_INDEX  
 INQ\_FILL\_AREA\_INTERIOR\_STYLE  
 INQ\_FILL\_AREA\_STYLE\_INDEX  
 INQ\_LINETYPE  
 INQ\_LINEWIDTH\_SCALE\_FACTOR  
 INQ\_LIST\_OF\_ASF  
 INQ\_POLYLINE\_COLOUR\_INDEX  
 INQ\_POLYMARKER\_COLOUR\_INDEX  
 INQ\_POLYMARKER\_SIZE\_SCALE\_FACTOR  
 INQ\_POLYMARKER\_TYPE  
 INQ\_TEXT\_COLOUR\_INDEX  
 INQ\_TEXT\_FONT\_AND\_PRECISION  
 INQ\_CURRENT\_NORMALIZATION\_TRANSFORMATION\_NUMBER  
 INQ\_CURRENT\_PRIMITIVE\_ATTRIBUTE\_VALUES

Функция ЯГС «Узнать значение атрибута текущего примитива вывода» отображается в следующие функции:

INQ\_POLYLINE\_INDEX  
 INQ\_POLYMARKER\_INDEX  
 INQ\_TEXT\_INDEX  
 INQ\_CHAR\_HEIGHT  
 INQ\_CHAR\_UP\_VECTOR  
 INQ\_CHAR\_WIDTH

INQ\_CHAR\_BASE\_VECTOR  
INQ\_TEXT\_PATH  
INQ\_TEXT\_ALIGNMENT  
INQ\_FILL\_AREA\_INDEX  
INQ\_PATTERN\_HEIGHT\_VECTOR  
INQ\_PATTERN\_WIDTH\_VECTOR  
INQ\_PATTERN\_REFERENCE\_POINT  
INQ\_DISPLAY\_SPACE\_SIZE  
INQ\_FILL\_AREA\_FACILITIES  
INQ\_GDP  
INQ\_LEVEL\_OF\_GKS  
INQ\_LIST\_OF\_AVAILABLE\_GDP  
INQ\_LIST\_OF\_AVAILABLE\_WS\_TYPES  
INQ\_LIST\_OF\_COLOUR\_INDICES  
INQ\_LIST\_OF\_NORMALIZATION\_TRANSFORMATION\_NUMBER  
INQ\_MAX\_LENGTH\_WS\_STATE\_TABLES  
INQ\_MAX\_NORMALIZATION\_TRANSFORMATION\_NUMBER  
INQ\_NORMALIZATION\_TRANSFORMATION  
INQ\_OPERATING\_STATE\_VALUE  
INQ\_PATTERN\_FACILITIES  
INQ\_PIXEL  
INQ\_PIXEL\_ARRAY  
INQ\_PIXEL\_ARRAY\_DIMENSIONS  
INQ\_POLYLINE\_FACILITIES  
INQ\_POLYMARKER\_FACILITIES  
INQ\_PREDEFINED\_COLOUR\_REPRESENTATION  
INQ\_PREDEFINED\_FILL\_AREA\_REPRESENTATION  
INQ\_PREDEFINED\_PATTERN\_REPRESENTATION  
INQ\_PREDEFINED\_POLYLINE\_REPRESENTATION  
INQ\_PREDEFINED\_POLYMARKER\_REPRESENTATION  
INQ\_PREDEFINED\_TEXT\_REPRESENTATION  
INQ\_SET\_OF\_OPEN\_WS  
INQ\_TEXT\_EXTENT  
INQ\_TEXT\_FACILITIES  
INQ\_WS\_CATEGORIES  
INQ\_WS\_CLASSIFICATION  
INQ\_WS\_CONNECTION\_AND\_TYPE  
INQ\_WS\_DEFERRAL\_AND\_UPDATE\_STATES  
INQ\_WS\_STATE  
INQ\_WS\_TRANSFORMATION  
INTERPRET\_ITEM  
OPEN\_GKS  
OPEN\_WS  
POLYLINE  
POLYMARKER  
READ\_ITEM\_FROM\_GKSM  
SELECT\_NORMALIZATION\_TRANSFORMATION  
SET\_ASF  
SET\_CHAR\_EXPANSION\_FACTOR  
SET\_CHAR\_HEIGHT

SET\_CHAR\_SPACING  
 SET\_CHAR\_UP\_VECTOR  
 SET\_CLIPPING\_INDICATOR  
 SET\_COLOUR\_REPRESENTATION  
 SET\_FILL\_AREA\_COLOUR\_INDEX  
 SET\_FILL\_AREA\_INDEX  
 SET\_FILL\_AREA\_INTERIOR\_STYLE  
 SET\_FILL\_AREA\_STYLE\_INDEX  
 SET\_LINETYPE  
 SET\_LINETHICKNESS\_SCALE\_FACTOR  
 SET\_MARKER\_SIZE\_SCALE\_FACTOR  
 SET\_MARKER\_TYPE  
 SET\_PATTERN\_REFERENCE\_POINT  
 SET\_PATTERN\_SIZE  
 SET\_POLYLINE\_COLOUR\_INDEX  
 SET\_POLYMARKER\_COLOUR\_INDEX  
 SET\_POLYLINE\_INDEX  
 SET\_POLYMARKER\_INDEX  
 SET\_TEXT\_ALIGNMENT  
 SET\_TEXT\_COLOUR\_INDEX  
 SET\_TEXT\_FONT\_AND\_PRECISION  
 SET\_TEXT\_INDEX  
 SET\_TEXT\_PATH  
 SET\_VIEWPORT  
 SET\_WINDOW  
 SET\_WS\_VIEWPORT  
 SET\_WS\_WINDOW  
 TEXT  
 UPDATE\_WS  
 WRITE\_ITEM\_TO\_GKSM

## Уровень 0b

INITIALISE\_CHOICE  
 INITIALISE\_LOCATOR  
 INITIALISE\_STRING  
 INITIALISE\_STROKE  
 INITIALISE\_VALUATOR  
 INQ\_CHOICE\_DEVICE\_STATE  
 INQ\_DEFAULT\_CHOICE\_DEVICE\_STATE  
 INQ\_DEFAULT\_LOCATOR\_DEVICE\_DATA  
 INQ\_DEFAULT\_STRING\_DEVICE\_DATA  
 INQ\_DEFAULT\_STROKE\_DEVICE\_DATA  
 INQ\_DEFAULT\_VALUATOR\_DEVICE\_DATA  
 INQ\_LOCATOR\_DEVICE\_STATE  
 INQ\_NUMBER\_OF\_AVAILABLE\_LOGICAL\_INPUT\_DEVICES  
 INQ\_STRING\_DEVICE\_STATE  
 INQ\_STROKE\_DEVICE\_STATE  
 INQ\_VALUATOR\_DEVICE\_STATE  
 REQUEST\_CHOICE  
 REQUEST\_LOCATOR

REQUEST\_STRING  
 REQUEST\_STROKE  
 REQUEST\_VALUATOR  
 SET\_CHOICE\_MODE  
 SET\_LOCATOR\_MODE  
 SET\_STRING\_MODE  
 SET\_STROKE\_MODE  
 SET\_VALUATOR\_MODE  
 SET\_VIEWPORT\_INPUT\_PRIORITY

## Уровень 0с

AWAIT\_EVENT  
 FLUSH\_DEVICE\_EVENTS  
 GET\_CHOICE  
 GET\_LOCATOR  
 GET\_STRING  
 GET\_STROKE  
 GET\_VALUATOR  
 INQ\_INPUT\_QUEUE\_OVERFLOW  
 INQ\_MORE\_SIMULTANEOUS\_EVENTS  
 SAMPLE\_CHOICE  
 SAMPLE\_LOCATOR  
 SAMPLE\_STRING  
 SAMPLE\_STROKE  
 SAMPLE\_VALUATOR

## Уровень 1а

ACCUMULATE\_TRANSFORMATION\_MATRIX  
 CLOSE\_SEGMENT  
 CREATE\_SEGMENT  
 DELETE\_SEGMENT  
 DELETE\_SEGMENT\_FROM\_WS  
 EVALUATE\_TRANSFORMATION\_MATRIX  
 INQ\_DEFAULT\_DEFERRAL\_STATE\_VALUES  
 INQ\_DYNAMIC\_MODIFICATION\_OF\_SEGMENT\_ATTRIBUTES  
 INQ\_DYNAMIC\_MODIFICATION\_OF\_WS\_ATTRIBUTES  
 INQ\_FILL\_AREA\_REPRESENTATION  
 INQ\_LIST\_OF\_FILL\_AREA\_INDICES  
 INQ\_LIST\_OF\_PATTERN\_INDICES  
 INQ\_LIST\_OF\_POLYLINE\_INDICES  
 INQ\_LIST\_OF\_POLYMARKER\_INDICES  
 INQ\_LIST\_OF\_TEXT\_INDICES  
 INQ\_NAME\_OF\_OPEN\_SEGMENT  
 INQ\_NUMBER\_OF\_SEGMENT\_PRIORITIES\_SUPPORTED  
 INQ\_PATTERN\_REPRESENTATION  
 INQ\_POLYLINE\_REPRESENTATION  
 INQ\_POLYMARKER\_REPRESENTATION

	INQ_SEGMENT_ATTRIBUTES INQ_SET_OF_ACTIVE_WS INQ_SET_OF_ASSOCIATED_WS INQ_SET_OF_SEGMENT_NAMES_IN_USE INQ_SET_OF_SEGMENT_NAMES_ON_WS INQ_TEXT_REPRESENTATION INQ_WS_MAX_NUMBERS MESSAGE REDRAW_ALL_SEGMENTS_ON_WS RENAME_SEGMENT SET_DEFERRAL_STATE SET_FILL_AREA_REPRESENTATION SET_HIGHLIGHTING SET_PATTERN_REPRESENTATION SET_POLYLINE_REPRESENTATION SET_POLYMARKER_REPRESENTATION SET_SEGMENT_PRIORITY SET_SEGMENT_TRANSFORMATION SET_TEXT_REPRESENTATION SET_VISIBILITY
Уровень 1b	INITIALISE_PICK INQ_CURRENT_PICK_ID_VALUE INQ_DEFAULT_PICK_DEVICE_DATA INQ_PICK_DEVICE_STATE REQUEST_PICK SET_DETECTABILITY SET_PICK_ID SET_PICK_MODE
Уровень 1c	GET_PICK SAMPLE_PICK
Уровень 2c	ASSOCIATE_SEGMENT_WITH_WS COPY_SEGMENT_TO_WS INSERT_SEGMENT
Уровень 2b	Отсутствуют
Уровень 2c	Отсутствуют

## 4.2. Определение типов данных

## 4.2.1. Сокращения, используемые в определениях типов данных

ASF	Флаг выборки атрибутов
CHAR	Литера
DC	Координата устройства
GDP	Обобщенный примитив вывода
GKS	Ядро графической системы
GKSM	Метафайл ядра графической системы
ID	Идентификатор
MAX	Максимум
NDC	Нормализованные координаты устройства
WC	Мировые координаты
WS	Станция

## 4.2.2. Определение типов в алфавитном порядке

В данном разделе в алфавитном порядке даются определения типов данных, используемые в связывании Ады с ЯГС. Каждая такая декларация задает уровень, на котором должна иметься декларация данных в реализации ЯГС данного уровня и любого более высокого уровня, в котором декларация типа впервые понадобится (аналогично функциям). Каждый декларируемый элемент также включает комментарий о типе и/или/использовании типа. Некоторые декларации в определении типа содержат константы. Все эти декларации констант включены в пакет GKS\_TYPE.

**ASF** Уровень 0a

type ASF is (BUNDLED, INDIVIDUAL);

Данный тип определяет флаг выборки атрибутов, чье значение указывает, должен ли атрибут примитива загружаться из таблицы связей или из индивидуального атрибута.

**ASF\_LIST** Уровень 0a

type ASF\_LIST is  
record

TYPE_OF_LINE_ASF	: ASF;
WIDTH_ASF	: ASF;
LINE_COLOUR_ASF	: ASF;
TYPE_OF_MARKER_ASF	: ASF;
SIZE_ASF	: ASF;

```

MARKER_COLOUR_ASF      : ASF;
FONT_PRECISION_ASF     : ASF;
EXPANSION_ASF          : ASF;
SPACING_ASF            : ASF;
TEXT_COLOUR_ASF        : ASF;
INTERIOR_ASF           : ASF;
STYLE_ASF               : ASF;
FILL_AREA_COLOUR_ASF   : ASF;
_ASF
end record;

```

Запись содержит все исходные флаги выборки атрибутов с компонентами, указанными индивидуальными флагами.

**ATTRIBUTES\_USED** Уровень 0a  
 package ATTRIBUTES\_USED is  
 new GKS\_LIST\_UTILITIES (ATTRIBUTES\_USED\_TYPE);  
 Предоставляет список используемых атрибутов.

**ATTRIBUTES\_USED\_TYPE** Уровень 0a  
 type ATTRIBUTES\_USED\_TYPE is (POLYLINE\_ATTRIBUTES,  
 POLYMARKER\_ATTRIBUTES,  
 TEXT\_ATTRIBUTES,  
 FILL\_AREA\_ATTRIBUTES);

Типы атрибутов, которые могут быть использованы в генерируемом выводе для GDP и в генерируемой информации подсказки и эха различных классов входных устройств.

**CHAR\_EXPANSION** Уровень 0a  
 type CHAR\_EXPANSION is new SCALE\_FACTOR range  
 SCALE\_FACTOR'SAFE\_SMALL .. SCALE\_FACTOR'LAST;  
 Определяет масштаб расширения литер. Масштаб должен быть больше нуля.

**CHAR\_SPACING** Уровень 0a  
 type CHAR\_SPACING is new SCALE\_FACTOR;  
 Определяет межлитерный просвет. Положительное значение межлитерного просвета в строке текста, а отрицательное значение обозначает перекрытие между прямоугольниками литер в строке текста.

**CHOICE\_DEVICE\_NUMBER** Уровень 0b  
 type CHOICE\_DEVICE\_NUMBER is new DEVICE\_NUMBER;  
 Обеспечивает идентификаторы устройств выбора.

- 
- CHOICE\_PROMPT** Уровень 0b  
 type CHOICE\_PROMPT is (OFF, ON);  
 Указывает на выбор типа подсказки и эха и будет или нет отображаться заданная подсказка.
- 
- CHOICE\_PROMPTS** Уровень 0b  
 package CHOICE\_PROMPT is  
 new GKS\_LIST\_UTILITIES (CHOICE\_PROMPT);  
 Предоставляет списки подсказок.
- 
- CHOICE\_PROMPT\_ECHO\_TYPE** Уровень 0b  
 type CHOICE\_PROMPT\_ECHO\_TYPE is new INTEGER;  
 Определяет выбранный тип подсказки и эха.
- 
- CHOICE\_PROMPT\_ECHO\_TYPES** Уровень 0b  
 package CHOICE\_PROMPT\_ECHO\_TYPES is  
 new GKS\_LIST\_UTILITIES (CHOICE\_PROMPT\_ECHO\_TYPE);  
 Предоставляет списки выбранной подсказки и эха.
- 
- CHOICE\_PROMPT\_STRING** Уровень 0b  
 type CHOICE\_PROMPT\_STRING (LENGTH:STRING\_SMALL..  
 NATURAL := 0) is  
 record  
 CONTENTS: STRING (1..LENGTH);  
 end record;  
 Предоставляется для подсказок переменной длины. Типы должны быть декларированы таким образом, чтобы была возможна динамическая модификация длины.
- 
- CHOICE\_PROMPT\_STRING\_ARRAY** Уровень 0b  
 type CHOICE\_PROMPT\_STRING\_ARRAY is array (POSITIVE  
 range<>)  
 of CHOICE\_PROMPT\_STRING;  
 Обеспечивает матрицу строк подсказок.
- 
- CHOICE\_PROMPT\_STRING\_LIST** Уровень 0b  
 type CHOICE\_PROMPT\_STRING\_LIST (LENGTH:CHOICE..  
 SMALL..NATURAL := 0)  
 is record  
 LIST:CHOICE\_PROMPT\_STRING\_ARRAY (1..LENGTH);  
 is record  
 Обеспечивает списки строк подсказок.

- 
- CHOICE\_REQUEST\_STATUS** Уровень 0b  
 type CHOICE\_REQUEST\_STATUS is (OK, NOCHOICE, NONE);  
 Определяет статус выбора входной операции для функции запроса.
- 
- CHOICE\_SMALL\_NATURAL** Уровень 0b  
 subtype CHOICE\_SMALL\_NATURAL  
 is NATURAL RANGE 0.. CHOICE\_SMALL\_NATURAL\_MAX;  
 Это декларация подтипа, которая допускает вызов объектов записей для типа CHOICE\_PROMPT\_STRING\_LIST без возникновения прерывания STORAGE\_ERROR.
- 
- CHOICE\_STATUS** Уровень 0b  
 subtype CHOICE\_STATUS is CHOICE\_REQUEST\_STATUS  
 range OK.. NOCHOICE;  
 Указывает, был ли сделан оператором выбор для функций «опросить», «получить» и «узнать».
- 
- CHOICE\_VALUE** Уровень 0b  
 type CHOICE\_VALUE is new POSITIVE;  
 Определяет выбор значений, существующих для реализации.
- 
- CLIPPING\_INDICATOR** Уровень 0a  
 type CLIPPING\_INDICATOR is (CLIP, NOCLIP);  
 Указывает, будет или нет выполняться отсечение.
- 
- COLOUR\_AVAILABLE** Уровень 0a  
 type COLOUR\_AVAILABLE is (COLOUR, MONOCHROME);  
 Указывает, существует ли вывод цвета на станции.
- 
- COLOUR\_INDEX** Уровень 0a  
 subtype COLOUR\_INDEX is PIXEL\_COLOUR\_INDEX  
 range 0.. PIXEL\_COLOUR\_INDEX\_LAST;  
 Указывает на тип в таблицах цвета.
- 
- COLOUR\_INDICES** Уровень 0a  
 package COLOUR\_INDICES is new  
 GKS\_LIST\_UTILITIES (COLOUR\_INDEX);  
 Предназначены для установки индикаторов цвета, которые имеются на конкретной станции.

**COLOUR\_MATRIX** Уровень 0a  
 type COLOUR\_MATRIX is array (POSITIVE range <>, POSITIVE range <>) of COLOUR\_INDEX;

Обеспечивает матрицы, содержащие индикаторы цвета, соответствующие матрице ячеек или матрице шаблонов.

**COLOUR\_REPRESENTATION** Уровень 0a  
 type COLOUR\_REPRESENTATION is record

RED : INTENSITY;

GREEN : INTENSITY;

BLUE : INTENSITY;

end record;

Определяет представление цвета как комбинацию интенсивностей в системе цветов красный—зеленый—голубой.

**CONTROL\_FLAG** Уровень 0a  
 type CONTROL\_FLAG is (CONDITIONALLY, ALWAYS);

Флаги управления используются для указания условий, при которых носитель изображения должен быть очищен.

**DC** Уровень 0a  
 package DC is new GKS\_COORDINATE\_SYSTEM (DC\_TYPE);  
 Определяет систему координат устройства.

**DC\_TYPE** Уровень 0a  
 type DC\_TYPE is digits PRECISION;  
 Тип координат в системе координат устройства.

**DC\_UNITS** Уровень 0a  
 type DC\_UNITS is (METRES, OTHER);

Единицы координат устройства для конкретной станции должны задаваться в метрах, если устройство не способно порождать точно масштабированные образы, или в соответствующих зависящих от станции единицах в противном случае.

**DEFERRAL\_MODE** Уровень 0a  
 type DEFERRAL\_MODE is (ASAP, BNIG, BNIL, ASTI);  
 Определяет четыре задержанных режима ЯГС.

**DEVICE\_NUMBER** Уровень 0b  
 package DEVICE\_NUMBER\_TYPE is  
 DEVICE\_NUMBER\_TYPE is new POSITIVE;

end DEVICE\_NUMBER\_TYPE;

К логическим устройствам входа обращаются по номерам устройств.

---

DISPLAY\_CLASS Уровень 0a  
 type DISPLAY\_CLASS is (VECTOR\_DISPLAY,  
 RASTER\_DISPLAY,  
 OTHER\_DISPLAY);

Классификация станций по категориям OUTPUT (вывод) или OUTIN (ввод/вывод).

---

DISPLAY\_SURFACE\_EMPTY Уровень 0a  
 type DISPLAY\_SURFACE\_EMPTY is (EMPTY, NOTEMPTY);  
 Указывает, очищен ли носитель изображения.

---

DYNAMIC\_MODIFICATION Уровень 1a  
 type DYNAMIC\_MODIFICATION is (IRG, IMM);  
 Указывает, обновление списка состояний выполняется немедленно или требуется повторная генерация.

---

ECHO\_SWITCH Уровень 0b  
 type ECHO\_SWITCH is (ECHO, NOECHO);  
 Указывает на то, выполняется или нет эхо-вывод подсказки.

---

ERROR\_NUMBER Уровень 0a  
 type ERROR\_NUMBER is new INTEGER;  
 Определяет тип для значений индикаторов ошибок.

---

EVENT\_DEVICE\_NUMBER Уровень 0c  
 type EVENT\_DEVICE\_NUMBER (CLASS : INPUT\_CLASS := NONE) is  
 record  
 case CLASS is  
 when NONE => null,  
 when LOCATOR\_INPUT => LOCATOR\_EVENT\_DEVICE\_NUMBER;  
 when STROKE\_INPUT => STROKE\_EVENT\_DEVICE\_NUMBER;  
 when VALUATOR\_INPUT => VALUATOR\_EVENT\_DEVICE\_NUMBER;

```

when CHOICE_INPUT      => CHOICE_EVENT_DEVICE
                        : CHOICE_DEVICE_
                          _NUMBER;
when PICK_INPUT        => PICK_EVENT_DEVICE
                        : PICK_DEVICE_
                          _NUMBER;
when STRING_INPUT     => STRING_EVENT_DEVICE
                        : STRING_DEVICE_
                          _NUMBER;

```

```
end case;
```

```
end record;
```

Обеспечивает возврат любого класса номера устройства из очереди событий.

EVENT\_OVERFLOW\_DEVICE\_NUMBER Уровень 0с

```
type EVENT_OVERFLOW_DEVICE_NUMBER
```

```
(CLASS : INPUT_QUEUE_CLASS := LOCATOR_INPUT) is
```

```
record
```

```
case CLASS is
```

```
when LOCATOR_INPUT    => LOCATOR_EVENT_
                        _DEVICE
                        : LOCATOR_DEVICE_
                          _NUMBER;
```

```
when STROKE_INPUT     => STROKE_EVENT_DEVICE
                        : STROKE_DEVICE_
                          _NUMBER;
```

```
when VALUATOR_
  _INPUT              => VALUATOR_EVENT_
                        _DEVICE
                        : VALUATOR_DEVICE_
                          _NUMBER;
```

```
when CHOICE_INPUT     => CHOICE_EVENT_
                        _DEVICE
                        : CHOICE_DEVICE_
                          _NUMBER;
```

```
when PICK_INPUT       => PICK_EVENT_DEVICE
                        : PICK_DEVICE_
                          _NUMBER;
```

```
when STRING_INPUT    => STRING_EVENT_DEVICE
                        : STRING_DEVICE_
                          _NUMBER;
```

```
end case;
```

```
end record;
```

Дается номер класса устройства для возврата для очереди событий.

<b>FILL_AREA_INDEX</b> type FILL_AREA_INDEX is new POSITIVE; Определяются индексы таблицы связей полигональной области.	Уровень 0a
<b>FILL_AREA_INDICES</b> package FILL_AREA_INDICES is new GKS_LIST_UTILITIES (FILL_AREA_INDEX); Обеспечивает список индексов таблицы связей полигональных областей.	Уровень 0a
<b>GDP_ID</b> type GDP_ID is new INTEGER; Осуществляет выбор среди различных обобщенных примитивов вывода.	Уровень 0a
<b>GDP_IDS</b> package GDP_IDS is new GKS_LIST_UTILITIES (GDP_ID); Предоставляет список идентификаторов обобщенных примитивов вывода.	Уровень 0a
<b>GKS_Level</b> type GKS_Level is (L0a, L0b, L0c, L1a, L1b, L1c, L2a, L2b, L2c); Имеющиеся уровни ЯГС.	Уровень 0a
<b>GKSM_ITEM_TYPE</b> type GKSM_ITEM_TYPE is new NATURAL; Тип элементов, содержащихся в метафайле GKSM.	Уровень 0a
<b>HATCH_STYLE</b> subtype HATCH_STYLE is STYLE_INDEX; Определяет тип варианта штриховки полигональной области.	Уровень 0a
<b>HATCH_STYLES</b> package HATCH_STYLES is new GKS_LIST_UTILITIES (HATCH_STYLE); Предоставляет список видов штриховки.	Уровень 0a
<b>HORIZONTAL_ALIGNMENT</b> type HORIZONTAL_ALIGNMENT is (NORMAL, LEFT, CENTRE, RIGHT); Выравнивание параллелограмма текста по отношению к горизонтальной позиции текста.	Уровень 0a

**IMPLEMENTATION\_DEFINED\_ERROR** Уровень 0a  
 subtype IMPLEMENTATION\_DEFINED\_ERROR is  
     **ERROR\_NUMBER**  
     range ERROR\_NUMBER'FIRST..-1;

Определяет область значений номеров ошибок, чтобы указать, какие ошибки, заданные в реализации, могут произойти.

**INDIVIDUAL\_ATTRIBUTE\_VALUES** Уровень 0a  
 type INDIVIDUAL\_ATTRIBUTE\_VALUES is

```
record
  TYPE_OF_LINE           : LINETYPE;
  WIDTH                  : LINEWIDTH;
  LINE_COLOUR            : COLOUR_INDEX;
  TYPE_OF_MARKER         : MARKER_TYPE;
  SIZE                   : MARKER_SIZE;
  MARKER_COLOUR          : COLOUR_INDEX;
  FONT_PRECISION         : TEXT_FONT_PRECISION;
  EXPANSION              : CHAR_EXPANSION;
  SPACING                : CHAR_SPACING;
  TEXT_COLOUR            : COLOUR_INDEX;
  INTERIOR               : INTERIOR_STYLE;
  STYLE                  : STYLE_INDEX;
  FILL_AREA_COLOUR       : COLOUR_INDEX;
  ASF                    : ASF_LIST;
end record;
```

Запись, содержащая все текущие индивидуальные атрибуты для процедуры INQ\_CURRENT\_INDIVIDUAL\_ATTRIBUTE\_VALUES.

**INPUT\_CLASS** Уровень 0b  
 type INPUT\_CLASS is (NONE,

```
    LOCATOR_INPUT,
    STROKE_INPUT,
    VALUATOR_INPUT,
    CHOICE_INPUT,
    PICK_INPUT,
    STRING_INPUT);
```

Определяет классификации входных устройств для рабочих станций категорий INPUT и OUTIN.

**INPUT\_QUEUE\_CLASS** Уровень 0c  
 subtype INPUT\_QUEUE\_CLASS is INPUT\_CLASS range  
     LOCATOR\_INPUT..STRING\_INPUT;

Определяет классификации для входных устройств для ситуаций, в которых отсутствие классификации невозможно.

**INPUT\_STATUS** Уровень 0b

type INPUT\_STATUS is (OK, NONE);

Определяет состояние устройства ввода позиции, ввода последовательности позиций, ввода числа и ввода строки.

**INPUT\_STRING** Уровень 0b

type INPUT\_STRING (LENGTH : STRING\_SMALL\_NATURAL  
:= 0) is

record

CONTENTS : STRING (1..LENGTH),

end record;

Предоставляет строку переменной длины. Объекты данного типа следует декларировать, чтобы позволить динамическую модификацию длины.

**INTENSITY** Уровень 0a

type INTENSITY is digits PRECISION range 0.0..1.0;

Определяет область возможных значений интенсивностей цвета.

**INTERIOR\_STYLE** Уровень 0a

type INTERIOR\_STYLE is (HOLLOW, SOLID, PATTERN,  
HATCH);

Определяет вид заполнения полигональных областей.

**INTERIOR\_STYLES** Уровень 0a

package INTERIOR\_STYLES is

new GKS\_LIST\_UTILITIES (INTERIOR\_STYLE);

Предоставляет список видов заполнения.

**INVALID\_VALUES\_INDICATOR** Уровень 0a

type INVALID\_VALUES\_INDICATOR is (ABSENT,  
PRESENT);

Указывает, присутствует или отсутствует значение  $-1$  в параметре PIXEL\_ARRAY, возвращаемом INQ\_PIXEL\_ARRAY.

**LANGUAGE\_BINDING\_ERROR** Уровень 0a

subtype LANGUAGE\_BINDING\_ERROR is ERROR\_NUMBER  
range 2500..2999;

Обозначает диапазон номеров ошибок для обозначения ошибок связки с языком, которые могут произойти.

LINETYPE	Уровень 0a
type LINETYPE is new INTEGER; Определяет типы линий, представляемых ЯГС.	
LINETYPES	Уровень 0a
package LINETYPES is new GKS_LIST_UTILITIES (LINETYPE);	
Предоставляет список типов линий.	
LINEWIDTH	Уровень 0a
type LINEWIDTH is new SCALE_FACTOR range 0.0 .. SCALE_FACTOR'LAST;	
Толщина линии определяется коэффициентом масштабирования.	
LOCATOR_DEVICE_NUMBER	Уровень 0b
type LOCATOR_DEVICE_NUMBER is new DEVICE_NUMBER; Определяет идентификаторы индикаторов местоположений устройства.	
LOCATOR_PROMPT_ECHO_TYPE	Уровень 0b
type LOCATOR_PROMPT_ECHO_TYPE is new INTEGER; Определяет подсказку определителя местоположения и типы эха, поддерживаемые реализацией.	
LOCATOR_PROMPT_ECHO_TYPES	Уровень 0b
package LOCATOR_PROMPT_ECHO_TYPES is new GKS_LIST_UTILITIES (LOCATOR_PROMPT_ECHO_ _TYPE);	
Предоставляет списки подсказок индикаторов местоположения и типов эха.	
MARKER_SIZE	Уровень 0a
type MARKER_SIZE is new SCALE_FACTOR range 0.0 .. SCALE_FACTOR'LAST;	
Размер маркера указывается масштабом.	
MARKER_TYPE	Уровень 0a
type MARKER_TYPE is new INTEGER; Определяет типы маркеров, представляемых ЯГС.	
MARKER_TYPES	Уровень 0a
package MARKER_TYPES is new GKS_LIST_UTILITIES	

## (MARKER\_TYPE);

Предоставляет список типов маркеров.

---

**MORE\_EVENTS** Уровень 0c  
 type MORE\_EVENTS is (NOMORE, MORE);

Указывает, содержатся ли события в очереди входных событий.

---

**NDC** Уровень 0a  
 package NDC is new GKS\_COORDINATE\_SYSTEM  
 (NDC\_TYPE);

Определяет нормализованную систему координат устройства.

---

**NDC\_TYPE** Уровень 0a  
 type NDC\_TYPE is digits PRECISION;

Определяет типы координат в нормализованной системе координат устройства.

---

**NEW\_FRAME\_NECESSARY** Уровень 0a  
 type NEW\_FRAME\_NECESSARY is (NO, YES);

Указывает, необходимы ли новые действия над кадром при модификации.

---

**OPERATING\_MODE** Уровень 0b  
 type OPERATING\_MODE is (REQUEST\_MODE,  
 SAMPLE\_MODE, EVENT\_MODE);

Определяет режимы работы входного устройства.

---

**OPERATING\_STATE** Уровень 0a  
 type OPERATING\_STATE is (GKCL, CKOP, WSOP, WSAC,  
 SGOP);

Определяет пять рабочих состояний ЯГС.

---

**PATTERN\_INDEX** Уровень 0a  
 subtype PATTERN\_INDEX is STYLE\_INDEX range 1..  
 STYLE\_INDEX'LAST;

Определяет область возможных значений индекса таблицы шаблонов.

---

**PATTERN\_INDICES** Уровень 0a  
 package PATTERN\_INDICES is  
 new GKS\_LIST\_UTILITIES (PATTERN\_INDEX);

Обеспечивает списки индексов шаблонов.

---

PICK_DEVICE_NUMBER	Уровень 1b
type PICK_DEVICE_NUMBER is new DEVICE_NUMBER;	
Предоставляется для выбранных устройств.	
PICK_ID	Уровень 1b
type PICK_ID is new POSITIVE;	
Определяет область значений идентификаторов выбора, доступных в данной реализации.	
PICK_IDS	Уровень 1b
package PICK_IDS is new GKS_LIST_UTILITIES (PICK_ID);	
Предоставляет список идентификаторов выбора.	
PICK_PROMPT_ECHO_TYPE	Уровень 0b
type PICK_PROMPT_ECHO_TYPE is new INTEGER;	
Определяет типы подсказки и эха устройства выбора.	
PICK_PROMPT_ECHO_TYPES	Уровень 0b
package PICK_PROMPT_ECHO_TYPES is	
new GKS_LIST_UTILITIES	
(PICK_PROMPT_ECHO_TYPE);	
Предоставляет списки типов подсказок и эха устройства выбора.	
PICK_REQUEST_STATUS	Уровень 1b
type PICK_REQUEST_STATUS is (OK, NOPICK, NONE);	
Определяет статус входных операций указания для функций запроса.	
PICK_STATUS	Уровень 1b
subtype PICK_STATUS is PICK_REQUEST_STATUS	
range OK..NOPICK;	
Определяет статус входных операций указания для всех функций.	
PIXEL_COLOUR_INDEX	Уровень 0a
type PIXEL_COLOUR_INDEX is new INTEGER range -1..	
INTEGER'LAST;	
Типы для цветов пикселей, где значение -1 обозначает недопустимый индекс цвета.	
PIXEL_COLOUR_MATRIX	Уровень 0a
type PIXEL_COLOUR_MATRIX is array (POSITIVE	

range <>, POSITIVE range <>)  
of PIXEL\_COLOUR\_INDEX,

Предоставляет матрицу цветов пикселей.

POLYLINE_INDEX	Уровень 0a
type POLYLINE_INDEX is new POSITIVE; Определяет область значений индексов ломаной.	
POLYLINE_INDICES	Уровень 0a
package POLYLINE_INDICES is new GKS_LIST_UTILITIES (POLYLINE_INDEX); Предоставляет списки индексов ломаной.	
POLYMARKER_INDEX	Уровень 0a
type POLYMARKER_INDEX is new POSITIVE; Определяет область значений индикаторов таблицы связей полимаркеров.	
POLYMARKER_INDICES	Уровень 0a
package POLYMARKER_INDICES is new GKS_LIST_UTILITIES (POLYMARKER_INDEX); Предоставляет списки индексов полимаркеров.	
POSITIVE_TRANSFORMATION_NUMBER	Уровень 0a
subtype POSITIVE_TRANSFORMATION_NUMBER is TRANSFORMATION_NUMBER range 1.. TRANSFORMATION_NUMBER'LAST; Номер преобразования нормирования, соответствующий устанавливаемому преобразованию.	
PRIMITIVE_ATTRIBUTE_VALUES	Уровень 0a
type PRIMITIVE_ATTRIBUTE_VALUES is record	
INDEX_POLYLINE	: POLYLINE_INDEX;
INDEX_POLYMARKER	: POLYMARKER_INDEX;
INDEX_TEXT	: TEXT_INDEX;
CHAR_HEIGHT	: WC_MAGNITUDE;
CHAR_UP	: WC_VECTOR;
CHAR_WIDTH	: WC_MAGNITUDE;
CHAR_BASE	: WC_VECTOR;
PATH	: TEXT_PATH;
ALIGNMENT	: TEXT_ALIGNMENT;
INDEX_FILL_AREA	: FILL_AREA_INDEX;

PATTERN\_WIDTH\_ : WC. VECTOR;  
 \_VECTOR  
 PATTERN\_HEIGHT\_ : WC. VECTOR;  
 VECTO  
 PATTERN\_REFE- : WC. POINT;  
 RENCE-PO

end record;

Запись, содержащая все атрибуты текущего примитива для процедуры «Запросить значения атрибутов текущего примитива».

RADIANS Уровень 1а  
 type RADIANS is digits PRECISION;

Величины, используемые при выполнении преобразований сегментов (угол вращения). Положительные значения обозначают вращение против часовой стрелки.

RANGE\_OF\_EXPRESIONS Уровень 0а  
 type RANGE\_OF\_EXPRESIONS is  
 record

MIN : CHAR\_EXPANSION;  
 MAX : CHAR\_EXPANSION;

end record;

Предоставляет область значений масштабов расширения литер.

RASTER\_UNITS Уровень 0а  
 type RASTER\_UNITS is new POSITIVE;  
 Определяет область значений элементов раstra.

RASTER\_UNIT\_SIZE Уровень 0а  
 type RASTER\_UNIT\_SIZE is  
 record

X : RASTER\_UNITS;

Y : RASTER\_UNITS;

end record;

Определяет размер экрана дисплея в растровых единицах на растровом устройстве.

REGENERATION\_MODE Уровень 0а  
 type REGENERATION\_MODE is (SUPPRESSED, ALLOWED);  
 Указывает, запрещена или разрешена неявная повторная генерация.

- RELATIVE\_PRIORITY** Уровень 0a  
 type RELATIVE\_PRIORITY is (HIGHER, LOWER);  
 Обозначает относительный приоритет между двумя преобразованиями нормирования.
- 
- RETURN\_VALUE\_TYPE** Уровень 0a  
 type RETURN\_VALUE\_TYPE is (SET, REALIZED);  
 Указывает на то, является ли возвращаемое значение таким, какое оно было установлено в программе, или как оно реализовано на устройстве.
- 
- SCALE\_FACTOR** Уровень 0a  
 package SCALE\_FACTOR\_TYPE is  
 type SCALE\_FACTOR is digits PRECISION;  
 end SCALE\_FACTOR\_TYPE;  
 Тип, используемый для масштаба.
- 
- SEGMENT\_DETECTABILITY** Уровень 1a  
 type SEGMENT\_DETECTABILITY is (UNDETECTABLE,  
 DETECTABLE);  
 Указывает на то, являются ли нет сегменты обнаруживаемыми.
- 
- SEGMENT\_HIGHLIGHTING** Уровень 1a  
 type SEGMENT\_HIGHLIGHTING is (NORMAL,  
 HIGHLIGHTED);  
 Указывает, является ли сегмент выделенным.
- 
- SEGMENT\_NAME** Уровень 1a  
 type SEGMENT\_NAME is new POSITIVE;  
 Определяет диапазон имен сегмента.
- 
- SEGMENT\_NAMES** Уровень 1a  
 package SEGMENT\_NAMES is new GKS\_LIST\_UTILITIES  
 (SEGMENT\_NAME);  
 Предоставляет список имен сегментов.
- 
- SEGMENT\_PRIORITY** Уровень 1a  
 type SEGMENT\_PRIORITY is digits PRECISION  
 range 0.0 . 1.0;  
 Определяет приоритет сегмента.
-

- 
- SEGMENT\_VISIBILITY** Уровень 1a  
 type SEGMENT\_VISIBILITY is (VISIBLE, INVISIBLE);  
 Определяет видимость сегмента.
- 
- SMALL\_NATURAL** Уровень 0a  
 subtype SMALL\_NATURAL is NATURAL range 0 ..  
 SMALL\_NATURAL\_MAX;  
 Это декларация подтипа, которая позволяет объектам быть  
 записями различных типов без возникновения прерываний  
 STORAGE\_ERROR
- 
- STRING\_DEVICE\_NUMBER** Уровень 0b  
 type STRING\_DEVICE\_NUMBER is new DEVICE\_NUMBER;  
 Предоставляет устройства ввода строки.
- 
- STRING\_PROMPT\_ECHO\_TYPE** Уровень 0b  
 type STRING\_PROMPT\_ECHO\_TYPE is new INTEGER;  
 Определяет типы подсказки и эха устройства ввода строки.
- 
- STRING\_PROMPT\_ECHO\_TYPES** Уровень 0b  
 package STRING\_PROMPT\_ECHO\_TYPES is new  
 GKS\_LIST\_UTILITIES (STRING\_PROMPT\_ECHO\_TYPE);  
 Предоставляет списки типов подсказок и эха устройства ввода  
 строки.
- 
- STRING\_SMALL\_NATURAL** Уровень 0a  
 subtype STRING\_SMALL\_NATURAL is NATURAL  
 range 0 .. STRING\_SMALL\_NATURAL\_MAX;  
 Это декларация подтипа, которая позволяет объектам быть  
 записями различных типов без возникновения прерываний  
 STORAGE\_ERROR.
- 
- STROKE\_DEVICE\_NUMBER** Уровень 0b  
 type STROKE\_DEVICE\_NUMBER is new DEVICE\_NUMBER;  
 Предоставляет номера устройств ввода последовательностей  
 позиций.
- 
- STROKE\_PROMPT\_ECHO\_TYPE** Уровень 0b  
 type STROKE\_PROMPT\_ECHO\_TYPE is new INTEGER;  
 Определяет типы подсказок и эха устройств ввода последовательностей  
 позиций.
-

- 
- STROKE\_PROMPT\_ECHO\_TYPES** Уровень 0b  
 package STROKE\_PROMPT\_ECHO\_TYPES is  
 new GKS\_LIST\_UTILITIES (STROKE\_PROMPT\_ECHO\_TYPE);
- Предоставляет списки типов подсказок и эха устройств ввода последовательностей позиций.
- 
- STYLE\_INDEX** Уровень 0a  
 type STYLE\_INDEX is new INTEGER;  
 Индекс вида — это либо HATCH\_STYLE, либо PATTERN\_STYLE.
- 
- TEXT\_ALIGNMENT** Уровень 0a  
 type TEXT\_ALIGNMENT is  
 record  
 HORIZONTAL : HORIZONTAL\_ALIGNMENT;  
 VERTICAL : VERTICAL\_ALIGNMENT;  
 end record;
- Тип атрибута, управляющего позиционированием параллелограмма текста по отношению к позиции текста, имеющего горизонтальную и вертикальную составляющие, как задано выше.
- 
- TEXT\_EXTERN\_PARALLELOGRAM** Уровень 0a  
 type TEXT\_EXTERN\_PARALLELOGRAM is  
 record  
 LOWER\_LEFT : WC\_POINT;  
 LOWER\_RIGHT : WC\_POINT;  
 UPPER\_RIGHT : WC\_POINT;  
 UPPER\_LEFT : WC\_POINT;  
 end record;
- Определяет угловые точки параллелограмма размера текста по отношению к вертикально позиционируемому тексту.
- 
- TEXT\_FONT** Уровень 0a  
 type TEXT\_FONT is new INTEGER;  
 Определяет типы шрифтов, предоставляемых реализацией.
- 
- TEXT\_FONT\_PRECISION** Уровень 0a  
 type TEXT\_FONT\_PRECISION is  
 record  
 FONT : TEXT\_FONT;  
 PRECISION : TEXT\_PRECISION;

end record;

Данный тип определяет запись, описывающую шрифт и точность текста.

**TEXT\_FONT\_PRECISIONS** Уровень 0a  
 package TEXT\_FONT\_PRECISIONS is  
   new GKS\_LIST\_UTILITIES (TEXT\_FONT\_

PRECISION);

Предоставляет списки пар, описывающих шрифт и точность текста.

**TEXT\_INDEX** Уровень 0a  
 type TEXT\_INDEX is new POSITIVE;

Определяет область значений индексов таблицы связей текстов.

**TEXT\_INDICES** Уровень 0a  
 package TEXT\_INDICES is new GKS\_LIST\_UTILITIES  
   (TEXT\_INDEX);

Предоставляет списки индексов текста.

**TEXT\_PATH** Уровень 0a  
 type TEXT\_PATH is (RIGHT, LEFT, UP, DOWN);

Направление, выбранное для строки текста.

**TEXT\_PRECISION** Уровень 0a  
 type TEXT\_PRECISION is (STRING\_PRECISION,  
   CHAR\_PRECISION,  
   STROKE\_PRECISION);

Точность, с которой появляется текст.

**TRANSFORMATION\_FACTOR** Уровень 1a  
 type TRANSFORMATION\_FACTOR is  
 record  
   X : NDC\_TYPE;  
   Y : NDC\_TYPE;  
 end record;

Масштаб, используемый в матрицах преобразований для выполнения преобразования сегментов.

**TRANSFORMATION\_MATRIX** Уровень 1a  
 type TRANSFORMATION\_MATRIX is array (1..2, 1..3) of  
 NDC\_TYPE;

Для преобразований сегментов, отображаемых внутри пространства НК.

---

TRANSFORMATION\_NUMBER Уровень 0a  
 type TRANSFORMATION\_NUMBER is new NATURAL;  
 Номер преобразования нормирования.

---

TRANSFORMATION\_PRIORITY\_ARRAY Уровень 0a  
 type TRANSFORMATION\_PRIORITY\_ARRAY is array  
 (POSITIVE range <>) of TRANSFORMATION\_  
NUMBER;  
 Тип для запоминания номеров преобразования.

---

TRANSFORMATION\_PRIORITY\_LIST Уровень 0a  
 type TRANSFORMATION\_PRIORITY\_LIST (LENGTH:  
SMALL\_NATURAL := 0) is  
 record  
 CONTENTS: TRANSFORMATION\_PRIORITY\_ARRAY  
(1..LENGTH);  
 end record;  
 Предоставляет список приоритетов номеров преобразований.

---

UPDATE\_REGENERATION\_FLAG Уровень 0a  
 type UPDATE\_REGENERATION\_FLAG is (PERFORM,  
POSTPONE);  
 Флаг, показывающий на действие повторной генерации на изображение.

---

UPDATE\_STATE Уровень 0a  
 type UPDATE\_STATE is (NOTPENDING, PENDING);  
 Указывает на то, что было запрошено, но еще не выполнено изменение преобразования для станции.

---

VALUATOR\_DEVICE\_NUMBER Уровень 0b  
 type VALUATOR\_DEVICE\_NUMBER is new  
DEVICE\_NUMBER;  
 Предоставляет идентификаторы устройств ввода числа.

---

VALUATOR\_INPUT\_VALUE Уровень 0b  
 type VALUATOR\_INPUT\_VALUE is digits PRECISION;  
 Определяет диапазон точности входных значений для реализации.

---

- 
- VALUATOR\_PROMPT\_ECHO\_TYPE** Уровень 0b  
 type VALUATOR\_PROMPT\_ECHO\_TYPE is new INTEGER;  
 Определяет возможный диапазон типов подсказок и эха для устройства ввода числа.
- 
- VALUATOR\_PROMPT\_ECHO\_TYPES** Уровень 0b  
 package VALUATOR\_PROMPT\_ECHO\_TYPES is  
 new GKS\_LIST\_UTILITIES (VALUATOR\_PROMPT\_ECHO\_TYPE);  
 Предоставляет список типов подсказок и эха для устройства ввода числа.
- 
- VARIABLE\_COLOUR\_MATRIX** Уровень 0a  
 type VARIABLE\_COLOUR\_MATRIX  
 (DX:SMALL\_NATURAL:=0;  
 DY:SMALL\_NATURAL:=0) is  
 record  
 MATRIX:COLOUR\_MATRIX (1..DX, 1..DY);  
 Предоставляет матрицы переменной длины, содержащие индексы цвета, соответствующие матрице ячеек или матрице шаблонов.
- 
- VARIABLE\_CONNECTION\_ID** Уровень 0a  
 type VARIABLE\_CONNECTION\_ID  
 (LENGTH:STRING\_SMALL\_NATURAL:=0) is  
 record  
 CONNECT:STRING (1..LENGTH);  
 end record;  
 Определяет идентификатор связи переменной длины для INQ\_WS\_CONNECTION\_AND\_TYPE.
- 
- VARIABLE\_PIXEL\_COLOUR\_MATRIX** Уровень 0a  
 type VARIABLE\_PIXEL\_COLOUR\_MATRIX  
 (DX:SMALL\_NATURAL:=0; DY:SMALL\_NATURAL:=0) is  
 record  
 MATRIX:PIXEL\_COLOUR\_MATRIX (1..DX, 1..DY);  
 end record;  
 Предоставляет матрицы переменного размера для цвета пикселей.
- 
- VERTICAL\_ALIGNMENT** Уровень 0a  
 type VERTICAL\_ALIGNMENT is (NORMAL, TOP, CAP, HALF,  
 BASE, BOTTOM);

Выравнивание параллелограмма текста по отношению к вертикальной позиции текста.

---

WC Уровень 0a  
 package WC is new GKS.COORDINATE\_SYSTEM (WC\_TYPE);  
 Определяет мировую систему координат.

---

WC\_TYPE Уровень 0a  
 type WC\_TYPE is digits PRECISION;  
 Определяет точность для типов мировых координат.

---

WS\_CATEGORY Уровень 0a  
 type WS\_CATEGORY is (OUTPUT, INPUT, OUTIN, WISS,  
 M0, MI);  
 Тип для категорий станций ЯГС.

---

WS\_ID Уровень 0a  
 type WS\_ID is new POSITIVE;  
 Определяет область значений идентификаторов станций.

---

WS\_IDS Уровень 0a  
 package WS\_IDS is new GKS\_LIST\_UTILITIES (WS\_ID);  
 Предназначен для списков идентификаторов станций.

---

WS\_STATE Уровень 0a  
 type WS\_STATE is (INACTIVE, ACTIVE);  
 Состояние станции.

---

WS\_TYPE Уровень 0a  
 type WS\_TYPE is new POSITIVE;  
 Диапазон значений, соответствующих правильным типам станций. Константы, определяющие имена для различных типов станций, должны быть обеспечены реализацией.

---

WS\_TYPES Уровень 0a  
 package WS\_TYPES is new GKS\_LIST\_UTILITIES  
 (WS\_TYPE);  
 Предназначена для списков типов станций.

#### 4.2.3. Список определений личных типов

В данном разделе дается в алфавитном порядке список определений личных типов, используемых для задания связывания

Ады и ЯГС. Каждая из этих деклараций определяет уровень ЯГС, на котором должна иметься декларация типа в реализации ЯГС данного или любого более высокого уровня, в котором декларация типа впервые понадобится (аналогично функциям). Все эти элементы являются декларациями типа PRIVATE языка Ада. Эти декларации включены в пакет ЯГС для того, чтобы дать возможность манипулировать личными типами.

**CHOICE\_DATA\_RECORD** Уровень 0b  
 type CHOICE\_DATA\_RECORD (PROMPT\_ECHO\_TYPE:  
   CHOICE\_PROMPT\_ECHO\_TYPE: = DEFAULT\_CHOICE)  
 is private;

Определяет запись для инициализации ввода выбора. Структура записи зависит от реализации. Так как это личный тип, компоненты записи могут быть найдены только через использование подпрограмм для манипулирования записями данных ввода (п. 5.2.1).

**GKSM\_DATA\_RECORD** Уровень 0a  
 type GKSM\_DATA\_RECORD (TYPE\_OF\_ITEM:  
   GKSM\_ITEM\_TYPE: = 0; LENGTH: NATURAL: = 0)  
 is private;

Запись данных для метафайла GKSM. Так как это личный тип, то до компонента записи можно добраться только через использование подпрограмм для манипулирования записями данных ввода (п. 5.2.1).

**LOCATOR\_DATA\_RECORD** Уровень 0b  
 type LOCATOR\_DATA\_RECORD (PROMPT\_ECHO\_TYPE:  
   LOCATOR\_PROMPT\_ECHO\_TYPE: = DEFAULT\_LOCATOR)  
 is private;

Определяет запись для инициализации ввода позиции. Структура записи задается при реализации. Так как это личный тип, то до компонента записи можно добраться только через использование подпрограмм для манипулирования записями данных ввода (п. 5.2.1).

**PICK\_DATA\_RECORD** Уровень 0b  
 type PICK\_DATA\_RECORD (PROMPT\_ECHO\_TYPE:  
   PICK\_PROMPT\_ECHO\_TYPE: = DEFAULT\_PICK) is private;

Определяет запись для инициализации ввода указания. Структура записи определяется в реализации. Так как это личный тип, то до компонента записи можно добраться, только используя под-

программы для манипулирования записями данных ввода (п. 5.2.1).

---

**STRING\_DATA\_RECORD** Уровень 0b  
 type STRING\_DATA\_RECORD (PROMPT\_ECHO\_TYPE:  
     STRING\_PROMPT\_ECHO\_TYPE:=DEFAULT\_STRING)  
 is private;

Определяет запись для инициализации ввода строки. Структура записи определяется в реализации. Так как это личный тип, то до компонента записи можно добраться, только используя подпрограммы для манипулирования записями данных ввода (п. 5.2.1).

---

**STROKE\_DATA\_RECORD** Уровень 0b  
 type STROKE\_DATA\_RECORD (PROMPT\_ECHO\_TYPE:  
     STROKE\_PROMPT\_ECHO\_TYPE:=DEFAULT\_STROKE)  
 is private;

Определяет запись для инициализации ввода последовательности позиций. Структура записи определяется в реализации. Так как это личный тип, то до компонента записи можно добраться, только используя подпрограммы для манипулирования записями данных ввода (п. 5.2.1).

---

**VALUATOR\_DATA\_RECORD** Уровень 0b  
 type VALUATOR\_DATA\_RECORD (PROMPT\_ECHO\_TYPE:  
     VALUATOR\_PROMPT\_ECHO\_TYPE:=DEFAULT\_VALUATOR)  
 is private;

Определяет запись для инициализации ввода числа. Структура записи определяется в реализации. Так как это личный тип, то до компонента записи можно добраться, только используя подпрограммы для манипулирования записями данных ввода (п. 5.2.1).

#### 4.2.4. Список деклараций констант

В данном разделе приведены декларации зависящих от реализации констант для задания типов Ада/ЯГС. Некоторые из констант используют для задания принимаемых по умолчанию значений параметров для процедур ЯГС, определяемых в разд. 5. В данном разделе приведены также константы, которые представляют стандартные значения, задаваемые для некоторых типов ЯГС/Ада.

Следующие константы определяют стандартные типы линий ЯГС:

SOLID\_LINE : constant LINETYPE := 1;  
 DASHED\_LINE : constant LINETYPE := 2;  
 DOTTED\_LINE : constant LINETYPE := 3;  
 DASHED-DOTTED : constant LINETYPE := 4;  
 \_LINE

Следующие константы определяют стандартные типы маркеров ЯГС:

DOT\_MARKER : constant MARKER\_TYPE := 1;  
 PLUS\_MARKER : constant MARKER\_TYPE := 2;  
 STAR\_MARKER : constant MARKER\_TYPE := 3;  
 ZERO\_MARKER : constant MARKER\_TYPE := 4;  
 X\_MARKER : constant MARKER\_TYPE := 5;

Следующие константы определяют стандартные типы подсказок и эха, поддерживаемые ЯГС:

DEFAULT\_LOCATOR : constant LOCATOR\_PROMPT\_ECHO\_TYPE := 1;  
 CROSS\_HAIR\_LOCATOR : constant LOCATOR\_PROMPT\_ECHO\_TYPE := 2;  
 TRACKING\_CROSS\_LOCATOR : constant LOCATOR\_PROMPT\_ECHO\_TYPE := 3;  
 RUBBER\_BAND\_LINE\_LOCATOR : constant LOCATOR\_PROMPT\_ECHO\_TYPE := 4;  
 RECTANGLE\_LOCATOR : constant LOCATOR\_PROMPT\_ECHO\_TYPE := 5;  
 DIGITAL\_LOCATOR : constant LOCATOR\_PROMPT\_ECHO\_TYPE := 6;  
 DEFAULT\_STROKE : constant STROKE\_PROMPT\_ECHO\_TYPE := 1;  
 DIGITAL\_STROKE : constant STROKE\_PROMPT\_ECHO\_TYPE := 2;  
 MARKER\_STROKE : constant STROKE\_PROMPT\_ECHO\_TYPE := 3;  
 LINE\_STROKE : constant STROKE\_PROMPT\_ECHO\_TYPE := 4;  
 DEFAULT\_VALUATOR : constant VALUATOR\_PROMPT\_ECHO\_TYPE := 1;  
 GRAPHICAL\_VALUATOR : constant VALUATOR\_PROMPT\_ECHO\_TYPE := 2;  
 DIGITAL\_VALUATOR : constant VALUATOR\_PROMPT\_ECHO\_TYPE := 3;  
 DEFAULT\_CHOICE : constant CHOICE\_PROMPT\_ECHO\_TYPE := 1;

PROMPT_ECHO_	: constant CHOICE_PROMPT_ECHO_	_TYPE := 2;
_CHOICE		
STRING_PROMPT_	: constant CHOICE_PROMPT_ECHO_	_TYPE := 3;
_CHOICE		
STRING_INPUT_	: constant CHOICE_PROMPT_ECHO_	_TYPE := 4;
_CHOICE		
SEGMENT_	: constant CHOICE_PROMPT_ECHO_	_TYPE := 5;
_CHOICE		
DEFAULT_STRING	: constant STRING_PROMPT_ECHO_	_TYPE := 1;
DEFAULT_PICK	: constant PICK_PROMPT_ECHO_	_TYPE := 1;
GROUP_HIGH_	: constant PICK_PROMPT_ECHO_	_TYPE := 2;
_LIGHT_PICK		
SEGMENT_	: constant PICK_PROMPT_ECHO_	_TYPE := 3;
_HIGHLIGHT_PICK		

Следующие константы используют для определения принимаемых по умолчанию значений параметров для процедур ЯГС, определяемых в разд. 5:

DEFAULT_MEMO-	: constant := implementation_defined;
RY_UNITS	
PRECISION	: constant := implementation_defined;
SMALL_NATURAL-	: constant := implementation_defined;
_MAX	
CHOICE_SMALL-	: constant := implementation_defined;
_NATURAL_MAX	
STRING_SMALL-	: constant := implementation_defined;
_NATURAL_MAX	
DEFAULT_ERROR-	: constant := implementation_defined;
_FILE	

Следующая строка задает прерывание GKS\_ERROR, определенное в п. 3.2.3:

GKS\_ERROR : exception;

#### 4.3. Коды ошибок

Связывание требует применения процедуры ERROR\_HANDLING для обработки любых ошибок, которые возникают в процедурах ЯГС, исключая процедуры запросов. Полное описание требований по обработке ошибок имеется в п. 3.2.3.

Функции запроса ЯГС не порождают прерываний. Вместо этого они возвращают параметр индикатора ошибок, который содер-

жит номер ошибки, которая обнаружена. Это согласуется с философией ЯГС, говорящей, что при запросе не возникает ошибок. Номера ошибок соответствуют номерам ошибок из приложения Б спецификации ЯГС плюс дополнительные ошибки, определенные в данном документе. Отметим, что различные известные ошибочные условия могут быть обнаружены вне контроля ЯГС благодаря природе языка Ада и могут привести к прерыванию при запросе.

### 13.1 Задание кодов ошибок

Стандарт ИСО 7942 дает отображение номеров ошибок для каждой функции ЯГС. Ряд известных ошибок ЯГС не может быть обнаружен реализацией ЯГС в Аде из-за свойств языка Ада, таких как строгость определения типов данных. Эти ошибки приведены в разделе о кодах устраненных ошибок.

В дополнение к определенным в ЯГС ошибкам могут существовать ошибки, задаваемые при реализации, и ошибки, определяемые связыванием

### IMPLEMENTATION\_DEFINED\_ERROR

Ошибки, задаваемые при реализации, описаны в Руководстве пользователя по реализации и имеют коды ошибок меньше нуля.

### LANGUAGE\_BINDING\_ERROR

Ошибки связывания с языком являются специфическими для связывания ЯГС с Адой. Номера ошибок с 2500 по 2999 зарезервированы для зависящих от связывания с языком Ада ошибок. Следующие ошибки определены данным связыванием для специфических ошибок связывания с языком:

2500 Неправильное использование записи данных ввода.

Когда происходят следующие ошибки, автоматически возникает непредопределенное прерывание в Аде.

2501 Неизвестная ошибка, произошедшая при обработке:

2502 Ошибка применения утилиты LIST GKS.

### 4.3.2. Коды устраняемых ошибок

Следующие ошибки ЯГС представлены отдельно из-за некоторых свойств языка Ада или их использования в данной связке; они никогда не могут произойти в данной реализации ЯГС. Ошибки могут быть обнаружены компилятором или во время выполнения вне области действия ЯГС

Коды ошибок, устраняемых функциями:

- 20 Неверно задан идентификатор станции
- 22 Неверно задан тип станции
- 65 Масштаб толщины линии меньше нуля

- 71 Масштаб маркера меньше нуля  
 77 Масштаб расширения литеры меньше или равен нулю  
 78 Высота литеры меньше или равна нулю  
 87 Значение размера шаблона не положительно  
 91 Неверно заданы размерности массива индексов цвета  
 92 Индекс цвета меньше нуля  
 96 Интенсивность цвета лежит вне диапазона от нуля до единицы  
 97 Неверно задан идентификатор указания  
 120 Неверно задано имя сегмента  
 126 Приоритет сегмента вне диапазона от нуля до единицы  
 151 Неверно задано время ожидания  
 166 Неверно задана максимальная длина записи данных

## 5. ФУНКЦИИ В АДЕ, СВЯЗАННЫЕ С ЯДРОМ ГРАФИЧЕСКОЙ СИСТЕМЫ

## 5.1. Функции ЯГС

- OPEN GKS  
 ОТКРЫТЬ GKS Уровень 0a  
 procedure OPEN\_GKS  
 (ERROR\_FILE : in STRING := DEFAULT\_ERROR\_FILE;  
 AMOUNT\_OF\_MEMORY : in NATURAL := DEFAULT\_MEMORY\_UNITS);
- 
- CLOSE GKS  
 ЗАКРЫТЬ ЯГС Уровень 0a  
 procedure CLOSE\_GKS
- 
- OPEN WORKSTATION  
 ОТКРЫТЬ СТАНЦИЮ Уровень 0a  
 procedure OPEN\_WS  
 (WS : in WS-ID;  
 CONNECTION : in STRING;  
 TYPE\_OF\_WS : in WS-TYPE);
- 
- CLOSE WORKSTATION  
 ЗАКРЫТЬ СТАНЦИЮ Уровень 0a  
 procedure CLOSE\_WS  
 (WS : in WS-ID);
- 
- ACTIVATE WORKSTATION  
 АКТИВИРОВАТЬ СТАНЦИЮ Уровень 0a  
 procedure ACTIVATE\_WS  
 (WS : in WS-ID);

DEACTIVATE WORKSTATION ДЕАКТИВИРОВАТЬ СТАНЦИЮ procedure DEACTIVATE_WS (WS : in WS_ID;	Уровень 0a
CLEAR WORKSTATION_ ОЧИСТИТЬ ИЗОБРАЖЕНИЕ НА СТАНЦИИ procedure CLEAR_WS (WS : in WS_ID; FLAG : in CONTROL_FLAG);	Уровень 0a
REDRAW ALL SEGMENTS ON WORKSTATION ПЕРЕРИСОВАТЬ ВСЕ СЕГМЕНТЫ НА СТАНЦИИ procedure REDRAW_ALL_SEGMENTS_ON_WS (WS : in WS_ID);	Уровень 1a
UPDATE WORKSTATION ОБНОВИТЬ ИЗОБРАЖЕНИЕ НА СТАНЦИИ procedure UPDATE_WS (WS : in WS_ID, REGENERATION : in UPDATE_REGENERATION_ _FLAG);	Уровень 1a
SET DEFERRAL STATE ЗАДАТЬ РЕЖИМ ЗАДЕРЖКИ procedure SET_DEFERRAL_STATE (WS : in WS_ID; DEFERRAL : in DEFERRAL_MODE; REGENERATION : in REGENERATION_MODE);	Уровень 1a
MESSAGE СООБЩЕНИЕ procedure MESSAGE (WS : in WS_ID; CONTENTS : in STRING);	Уровень 1a
ESCAPE РАСШИРЕНИЕ	Уровень 0a

Функции расширения рассматриваются в данном связывании как отдельные процедуры для каждого типа расширения, предоставляемого реализацией, каждая со списком формальных пара-

метров, соответствующих реализованной процедуре. Зарегистрированные процедуры ESCAPE будут находиться в библиотечном пакете, названном GKS\_ESCAPE. Имена и параметры ESCAPE зарегистрированы в Международном журнале графических элементов ИСО, который ведется Органом регистрации.

Каждая незарегистрированная процедура ESCAPE будет находиться в библиотечном пакете; при этом используются следующие соглашения по именованию:

```
package GKS_UESC_<имя процедуры расширения> is
  procedure ESC;
```

— код на Аде для процедуры UESC

```
end GKS_UESC_<имя процедуры расширения>;
```

— Единственным именем процедуры, используемым в пакете, будет ESC

Для того, чтобы поддержать возможность записывать ESCAPE в метафайл, эти зарегистрированные расширения могут быть привлечены, используя типы данных и форму процедуры GENERALIZED\_ESC, которая имеет спецификацию, приведенную ниже:

```
package GKS_ESCAPE is
```

```
  type ESCAPE_ID is new INTEGER;
```

```
  type ESCAPE_FLOAT is digits PRECISION;
```

```
  type ESC_INTEGER_ARRAY is array (SMALL_NATURAL
    range <>) of INTEGER;
```

```
  type ESC_FLOAT_ARRAY is array (SMALL_NATURAL
    range <>) of ESCAPE_FLOAT;
```

```
  type ESC_STRING_ARRAY is array (SMALL_NATURAL
    range <>) of STRING (1..80);
```

```
  type ESC_DATA_RECORD (NUM_OF_INTEGER
```

```
    : SMALL_NATURAL := 0;
```

```
    : SMALL_NATURAL := 0;
```

```
    : SMALL_NATURAL := 0) is
```

```
    record
```

```
      INTEGER_ARRAY : ESC_INTEGER_ARRAY
```

```
        (1..NUM_OF_INTEGERS);
```

```
      REAL_ARRAY : ESC_FLOAT_ARRAY (1..NUM_OF_REALS);
```

```
      ESC_STRINGS : ESC_STRING_ARRAY
```

```
        (1..NUM_OF_STRINGS);
```

```
    end record;
```

```
  procedure GENERALIZED_ESC
```

```
    (ESCAPE_NAME : in ESCAPE_ID;
```

```
     ESC_DATA_IN : in ESC_DATA_RECORD;
```

```
     ESC_DATA_OUT : in ESC_DATA_RECORD);
```

end GKS\_ESCAPE;

Представляет типы данных и процедуры для реализации не-поддерживаемых расширений.

---

POLYLINE Уровень 0a  
 ЛОМАНАЯ  
 procedure POLYLINE  
 (POINTS : in WC\_POINT\_ARRAY);

---

POLYMARKER Уровень 0a  
 ПОЛИМАРКЕР  
 procedure POLYMARKER  
 (POINTS : in WC\_POINT\_ARRAY);

---

TEXT Уровень 0a  
 ТЕКСТ  
 procedure TEXT  
 (POSITION : in WC\_POINT;  
 CHAR\_STRING : in STRING);

---

FILL\_AREA Уровень 0a  
 ПОЛИГОНАЛЬНАЯ ОБЛАСТЬ  
 procedure FILL\_AREA  
 (POINTS : in WC\_POINT\_ARRAY);

---

CELL\_ARRAY Уровень 0a  
 МАТРИЦА ЯЧЕЕК  
 procedure CELL\_ARRAY  
 (CORNER\_1\_1 : in WC\_POINT;  
 CORNER\_DX DY : in WC\_POINT;  
 CELLS : in COLOUR\_MATRIX);

---

GENERALIZED DRAWING PRIMITIVE Уровень 0a  
 ОБОБЩЕННЫЙ ПРИМИТИВ ВЫВОДА

Обобщенный графический примитив вывода ОПВ связывается по принципу один во многие с отдельной процедурой, реализованной для каждого ОПВ, каждая из которых имеет свой собственный интерфейс. Зарегистрированные ОПВ находятся в библиотечном пакете, названном GKS\_GDR. Имена ОПВ и параметры зарегистрированы в Международном журнале графических записей ИСО, который ведется Органом регистрации.

Каждая незарегистрированная процедура GDP будет находиться в библиотечном пакете, использующем следующие правила именованья:

```
package GKS_UGDP_<имя процедуры GDP>
```

```
  procedure GDP;
```

```
    — код на Аде процедуры UGDP
```

```
end GKS_UGDP_<имя процедуры GDP>;
```

— Единственным именем процедуры, используемым в пакете, будет GDP

Для того, чтобы поддержать возможность записывать ОПВ данной реализации в метафайл, эти зарегистрированные ОПВ могут быть привлечены, используя типы данных и форму процедуры GENERALIZED\_GDP, которая имеет следующую спецификацию:

```
package GKS_GDP is
```

```
  type GDP_FLOAT is digits PRECISION;
```

```
  type GDP_INTEGER_ARRAY is array (SMALL_NATURAL  
  range <>) of INTEGER;
```

```
  type GDP_FLOAT_ARRAY is array (SMALL_NATURAL  
  range <>) of GDP_FLOAT;
```

```
  type GDP_STRING_ARRAY is array (SMALL_NATURAL  
  range <>) of STRING (1..80);
```

```
  type GDP_DATA_RECORD (NUM_OF_INTEGERS : SMALL_
    _NATURAL : =0;
    NUM_OF_REAL : SMALL_
    _NATURAL : =0;
    NUM_OF_STRINGS : SMALL_
    _NATURAL : =0)
```

```
  is
```

```
  record
```

```
    INTEGER_ARRAY : GDP_INTEGER_ARRAY (1.. NUM_OF_
      _INTEGERS);
```

```
    REAL_ARRAY : GDP_FLOAT_ARRAY (1.. NUM_OF_
      _REALS);
```

```
    GDP_STRINGS : GDP_STRING_ARRAY (1.. NUM_OF_
      _STRINGS);
```

```
  end record;
```

```
  procedure GENERALIZED_GDP (GDP_NAME : in GDP_ID;
    POINTS : in WC.POINT_LIST;
    GDP_DATA : in GDP_DATA_RECORD);
```

```
end GKS_GDP;
```

Предоставляет типы данных и процедуру для реализации неподдержанных обобщенных примитивов вывода.

SET POLYLINE INDEX ЗАДАТЬ ИНДЕКС ЛОМАННОЙ	Уровень 0a
procedure SET_POLYLINE_INDEX (INDEX : in POLYLINE_INDEX);	
SET LINETYPE ЗАДАТЬ ТИП ЛИНИИ	Уровень 0a
procedure SET_LINETYPE (TYPE_OF_LINE : in LINETYPE);	
SET LINEWIDTH SCALE FACTOR ЗАДАТЬ МАСШТАБ ТОЛЩИНЫ ЛИНИИ	Уровень 0a
procedure SET_LINEWIDTH_SCALE_FACTOR (WIDTH : in LINEWIDTH);	
SET POLYLINE COLOUR INDEX ЗАДАТЬ ИНДЕКС ЦВЕТА ЛОМАННОЙ	Уровень 0a
procedure SET_POLYLINE_COLOUR_INDEX (LINE_COLOUR : in COLOUR_INDEX);	
SET POLYMARKER INDEX ЗАДАТЬ ИНДЕКС ПОЛИМАРКЕРА	Уровень 0a
procedure SET_POLYMARKER_INDEX (INDEX : in POLYMARKER_INDEX);	
SET MARKER TYPE ЗАДАТЬ ТИП МАРКЕРА	Уровень 0a
procedure SET_MARKER_TYPE (TYPE)OF_MARKER : in MARKER_TYPE);	
SET MARKER SIZE SCALE FACTOR ЗАДАТЬ МАСШТАБ МАРКЕРА	Уровень 0a
procedure SET_MARKER_SIZE_SCALE_FACTOR (SIZE : in MARKER_SIZE);	
SET POLYMARKER COLOUR INDEX ЗАДАТЬ ИНДЕКС ЦВЕТА ПОЛУМАРКЕРА	Уровень 0a
procedure SET_POLYMARKER_COLOUR_INDEX (MARKER_COLOUR : in COLOUR_INDEX);	
SET TEXT INDEX ЗАДАТЬ ИНДЕКС ТЕКСТА	Уровень 0a
procedure SET_TEXT_INDEX (INDEX : in TEXT_INDEX);	

SET TEXT FONT AND_PRECISION ЗАДАТЬ ШРИФТ И ТОЧНОСТЬ ТЕКСТА procedure SET_TEXT_FONT_AND_PRECISION (FONT_PRECISION : in TEXT_FONT_PRECISION);	Уровень 0a
SET CHARACTER EXPANSION FACTOR ЗАДАТЬ МАСШТАБ РАСШИРЕНИЯ ЛИТЕРЫ procedure SET_CHAR_EXPANSION_FACTOR (EXPANSION : in CHAR_EXPANSION);	Уровень 0a
SET CHARACTER SPACING ЗАДАТЬ МЕЖЛИТЕРНЫЙ ПРОСВЕТ procedure SET_CHAR_SPACING (SPACING : in CHAR_SPACING);	Уровень 0a
SET TEXT COLOUR INDEX ЗАДАТЬ ИНДЕКС ЦВЕТА ТЕКСТА procedure SET_TEXT_COLOUR_INDEX (TEXT_COLOUR : in COLOUR_INDEX);	Уровень 0a
SET CHARACTER HEIGHT ЗАДАТЬ ВЫСОТУ ЛИТЕР procedure SET_CHAR_HEIGHT (HEIGHT : in WC_MAGNITUDE);	Уровень 0a
SET CHARACTER UP VECTOR ЗАДАТЬ ВЕРТИКАЛЬ ЛИТЕРЫ procedure SET_CHAR_UP_VECTOR (CHAR_UP_VECTOR : in WC_VECTOR);	Уровень 0a
SET TEXT PATH ЗАДАТЬ НАПРАВЛЕНИЕ ТЕКСТА procedure SET_TEXT_PATH (PATH : in TEXT_PATH);	Уровень 0a
SE TEXT ALIGNMENT ЗАДАТЬ ВЫРАВНИВАНИЕ ТЕКСТА procedure SET_TEXT_ALIGNMENT (ALIGNMENT : in TEXT_ALIGNMENT);	Уровень 0a
SET FILL AREA INDEX ЗАДАТЬ ИНДЕКС ПОЛИГОНАЛЬНОЙ ОБЛАСТИ procedure SET_FILL_AREA_INDEX (INDEX : in FILL_AREA_INDEX);	Уровень 0a

SET FILL AREA INTERIOR STYLE ЗАДАТЬ ВИД ЗАПОЛНЕНИЯ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ	Уровень 0a
procedure SET_FILL_AREA_INTERIOR_STYLE (INTERIOR : in INTERIOR_STYLE);	
SET FILL AREA STYLE INDEX ЗАДАТЬ ИНДЕКС ЗАПОЛНЕНИЯ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ	Уровень 0a
procedure SET_FILL_AREA_STYLE_INDEX (STYLE : in STYLE_INDEX);	
SET FILL AREA COLOUR INDEX ЗАДАТЬ ИНДЕКС ЦВЕТА ПОЛИГОНАЛЬНОЙ ОБЛАСТИ	Уровень 0a
procedure SET_FILL_AREA_COLOUR_INDEX (FILL_AREA_COLOUR : in COLOUR_INDEX);	
SET PATTERN SIZE ЗАДАТЬ РАЗМЕР ШАБЛОНА	Уровень 0a
procedure SET_PATTERN_SIZE (SIZE : in WC_SIZE);	
SET PATTERN REFERENCE POINT ЗАДАТЬ ТОЧКУ ПРИВЯЗКИ ШАБЛОНА	Уровень 0a
procedure SET_PATTERN_REFERENCE_POINT (POINT : in WC_POINT);	
SET ASPECTSOURCE FLAGS ЗАДАТЬ ФЛАГИ ВЫБОРКИ АТТРИБУТОВ	Уровень 0a
procedure SET_ASF (ASF : in ASF_LIST);	
SET PICK IDENTIFIER ЗАДАТЬ ИДЕНТИФИКАТОР УКАЗАНИЯ	Уровень 1b
procedure SET_PICK_ID (PICK : in PICK_ID);	
SET POLYLINE REPRESENTATION ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ЛОМАНОЙ	Уровень 1a
procedure SET_POLYLINE_REPRESENTATION (WS : in WS_ID; INDEX : in POLYLINE_INDEX; TYPE_OF_LINE : in LINETYPE;	

WIDTH : in LINEWIDTH;  
 LINE\_COLOUR : in COLOUR\_INDEX);

SET POLYMARKER REPRESENTATION Уровень 1a  
 ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИМАРКЕРА

procedure SET\_POLYMARKER\_REPRESENTATION  
 (WS : in WS\_ID;  
 INDEX : in POLYMARKER\_INDEX;  
 TYPE\_OF\_MARKER : in MARKER\_TYPE;  
 SIZE : in MARKER\_SIZE;  
 MARKER\_COLOUR : in COLOUR\_INDEX);

SET TEXT REPRESENTATION Уровень 1a  
 ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ТЕКСТА

procedure SET\_TEXT\_REPRESENTATION  
 (WS : in WS\_ID;  
 INDEX : in TEXT\_INDEX;  
 FONT\_PRECISION : in TEXT\_FONT\_PRECISION;  
 EXPANSION : in CHAR\_EXPANSION;  
 SPACING : in CHAR\_SPACING;  
 TEXT\_COLOUR : in COLOUR\_INDEX);

SET FILL AREA REPRESENTATION Уровень 1a  
 ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ

procedure SET\_FILL\_AREA\_REPRESENTATION  
 (WS : in WS\_ID;  
 INDEX : in FILL\_AREA\_INDEX;  
 INTERIOR : in INTERIOR\_STYLE;  
 STYLE : in STYLE\_INDEX;  
 FILL\_AREA\_COLOUR : in COLOUR\_INDEX);

SET PATTERN REPRESENTATION Уровень 1a  
 ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ШАБЛОНА

procedure SET\_PATTERN\_REPRESENTATION  
 (WS : in WS\_ID;  
 INDEX : in PATTERN\_INDEX;  
 PATTERN : in COLOUR\_MATRIX);

SET COLOUR REPRESENTATION Уровень 0a  
 ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ЦВЕТА

procedure SET\_COLOUR\_REPRESENTATION  
 (WS : in WS\_ID);

INDEX	: in COLOUR INDEX;	
RGB_COLOUR	: in COLOUR-REPRESENTATION);	
SET WINDOW		Уровень 0a
ЗАДАТЬ ОКНО		
procedure SET_WINDOW		
(TRANSFORMATION	: in POSITIVE_TRANSFORMATION_NUMBER;	
WINDOW_LIMITS	: in WC_RECTANGLE_LIMITS);	
SET VIEWPORT		Уровень 0a
ЗАДАТЬ ПОЛЕ ВЫВОДА		
procedure SET_VIEWPORT		
(TRANSFORMATION	: in POSITIVE_TRANSFORMATION_NUMBER;	
VIEWPORT_LIMITS	: in NDC_RECTANGLE_LIMITS);	
SET VIEWPORT INPUT PRIORITY		Уровень 0b
ЗАДАТЬ ПРИОРИТЕТ ПОЛЯ ВЫВОДА ПРИ ВВОДЕ		
procedure SET_VIEWPORT_INPUT_PRIORITY		
(TRANSFORMATION	: in TRANSFORMATION_NUMBER;	
REFERENCE_TRANSFORMATION	: in TRANSFORMATION_NUMBER;	
PRIORITY	: in RELATIVE_PRIORITY);	
SELECT NORMALIZATION		Уровень 0a
TRANSFORMATION		
ВЫБРАТЬ ПРЕОБРАЗОВАНИЕ НОРМИРОВАНИЯ		
procedure SELECT_NORMALIZATION_TRANSFORMATION		
(TRANSFORMATION	: in TRANSFORMATION_NUMBER;	
SET CLIPPING INDICATOR		Уровень 0a
ЗАДАТЬ ИНДИКАТОР ОТСЕЧЕНИЯ		
procedure SET_CLIPPING_INDICATOR		
(CLIPPING	: in CLIPPING_INDICATOR);	
SET WORKSTATION WINDOW		Уровень 0a
ЗАДАТЬ ОКНО СТАНЦИИ		
procedure SET_WS_WINDOW		

(WS : in WS\_ID;  
 WS\_WINDOW\_LIMITS : in NDC.RECTANGLE\_LIMITS);

SET WORKSTATION VIEWPORT Уровень 0a  
 ЗАДАТЬ ПОЛЕ ВЫВОДА СТАНЦИИ

procedure SET\_WS\_VIEWPORT  
 (WS : in WS\_ID;  
 WS\_VIEWPORT\_LIMITS : in DC.RECTANGLE\_LIMITS);

CREATE SEGMENT Уровень 1a  
 СОЗДАТЬ СЕГМЕНТ

procedure CREATE\_SEGMENT  
 (SEGMENT : in SEGMENT\_NAME);

CLOSE SEGMENT Уровень 1a  
 ЗАКРЫТЬ СЕГМЕНТ

procedure CLOUSE\_SEGMENT;

RENAME SEGMENT Уровень 1a  
 ПЕРЕИМЕНОВАТЬ СЕГМЕНТ

procedure RENAME\_SEGMENT  
 (OLD\_NAME : in SEGMENT\_NAME;  
 NEW\_NAME : in SEGMENT\_NAME);

DELETE SEGMENT Уровень 1a  
 УНИЧТОЖИТЬ СЕГМЕНТ

procedure DELETE\_SEGMENT  
 (SEGMENT : in SEGMENT\_NAME);

DELETE SEGMENT FROM Уровень 1a  
 WORKSTATION

УДАЛИТЬ СЕГМЕНТ СО СТАНЦИИ  
 procedure DELETE\_SEGMENT\_FROM\_WS  
 (WS : in WS\_ID;  
 SEGMENT : in SEGMENT\_NAME);

ASSOCIATE SEGMENT WITH Уровень 2a  
 WORKSTATION

СВЯЗАТЬ СЕГМЕНТ СО СТАНЦИЕЙ  
 procedure ASSOCIATE\_SEGMENT\_WITH\_WS  
 (WS : in WS\_ID;  
 SEGMENT : in SEGMENT\_NAME);

COPY SEGMENT TO WORKSTATION ВЫВЕСТИ КОПИЮ СЕГМЕНТА НА СТАНЦИЮ	Уровень 2a
procedure COPY_SEGMENT_TO_WS (WS : in WS_ID; SEGMENT : in SEGMENT_NAME);	
INSERT SEGMENT ВСТАВИТЬ СЕГМЕНТ	Уровень 2a
procedure INSERT_SEGMENT (SEGMENT : in SEGMENT_NAME; TRANSFORMATION : in TRANSFORMATION_ _MATRIX);	
SET SEGMENT TRANSFORMATION ЗАДАТЬ ПРЕОБРАЗОВАНИЕ СЕГМЕНТА	Уровень 1a
procedure SET_SEGMENT_TRANSFORMATION (SEGMENT : in SEGMENT_NAME; TRANSFORMATION : in TRANSFORMATION_ _MATRIX);	
SET VISIBILITY ЗАДАТЬ ВИДИМОСТЬ	Уровень 1a
procedure SET_VISIBILITY (SEGMENT : in SEGMENT_NAME; VISIBILITY : in SEGMENT_VISIBILITY);	
SET HIGHLIGHTING ЗАДАТЬ ВЫДЕЛЕНИЕ	Уровень 1a
procedure SET_HIGHLIGHTING (SEGMENT : in SEGMENT_NAME; HIGHLIGHTING : in SEGMENT_ _HIGHLIGHTING);	
SET SEGMENT PRIORITY ЗАДАТЬ ПРИОРИТЕТ СЕГМЕНТА	Уровень 1a
procedure SET_SEGMENT_PRIORITY (SEGMENT : in SEGMENT_NAME; PRIORITY : in SEGMENT_PRIORITY);	
SET DETECTABILITY ЗАДАТЬ ЧУВСТВИТЕЛЬНОСТЬ К УКАЗАНИЮ	Уровень 1b
procedure SET_DETECTABILITY	

(SEGMENT : in SEGMENT\_NAME;  
 DETECTABILITY : in SEGMENT\_DETECTABILITY);

INITIALISE LOCATOR Уровень 0b  
 ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТВО ПОЗИЦИИ  
 procedure INITIALISE\_LOCATOR  
 (WS : in WS\_ID;  
 DEVICE : in LOCATOR\_DEVICE\_NUMBER;  
 INITIAL\_TRANSFORMATION : in TRANSFORMATION\_NUMBER;  
 INITIAL\_POSITION : in WC\_POINT;  
 ECHO\_AREA : in DC\_RECTANGLE\_LIMITS;  
 DATA\_RECORD : in LOCATOR\_DATA\_RECORD);

INITIALISE STROKE Уровень 0b  
 ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТВО ВВОДА  
 ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИИ  
 procedure INITIALISE\_STROKE  
 (WS : in WS\_ID;  
 DEVICE : in STROKE\_DEVICE\_NUMBER;  
 INITIAL\_TRANSFORMATION : in TRANSFORMATION\_NUMBER;  
 INITIAL\_STROKE : in WC\_POINT\_ARRAY;  
 ECHO\_AREA : in DC\_RECTANGLE\_LIMITS;  
 DATA\_RECORD : in STROKE\_DATA\_RECORD);

INITIALISE VALUATOR Уровень 0b  
 ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТВО ВВОДА ЧИСЛА  
 procedure INITIALISE\_VALUATOR  
 (WS : in WS\_ID;  
 DEVICE : in VALUATOR\_DEVICE\_NUMBER;  
 INITIAL\_VALUE : in VALUATOR\_INPUT\_VALUE;  
 ECHO\_AREA : in DC\_RECTANGLE\_LIMITS;  
 DATA\_RECORD : in VALUATOR\_DATA\_RECORD);

INITIALISE CHOICE Уровень 0b  
 ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТВО ВЫБОРА

## procedure INITIALISE\_CHOICE

(WS : in WS\_ID;  
 DEVICE : in CHOICE\_DEVICE\_NUMBER;  
 INITIAL\_STATUS : in CHOICE\_STATUS;  
 INITIAL\_CHOICE : in CHOICE\_VALUE;  
 ECHO\_AREA : in DC\_RECTANGLE\_LIMITS;  
 DATA\_RECORD : in CHOICE\_DATA\_RECORD);

## INITIALISE PICK

Уровень 1b

## ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТВО УКАЗАНИЯ

## procedure INITIALISE\_PICK

(WS : in WS\_ID;  
 DEVICE : in PICK\_DEVICE\_NUMBER;  
 INITIAL\_STATUS : in PICK\_STATUS;  
 INITIAL\_SEGMENT : in SEGMENT\_NAME;  
 INITIAL\_PICK : in PICK\_ID;  
 ECHO\_AREA : in DC\_RECTANGLE\_LIMITS;  
 DATA\_RECORD : in PICK\_DATA\_RECORD);

## INITIALISE STRING

Уровень 0b

## ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТВО ВВОДА СТРОКИ

## procedure INITIALIZE\_STRING

(WS : in WS\_ID;  
 DEVICE : in STRING\_DEVICE\_NUMBER;  
 INITIAL\_STRING : in INPUT\_STRING;  
 ECHO\_AREA : in DC\_RECTANGLE\_LIMITS;  
 DATA\_RECORD : in STRING\_DATA\_RECORD);

## SET LOCATOR MODE

Уровень 0b

## ЗАДАТЬ РЕЖИМ УСТРОЙСТВА ВВОДА ПОЗИЦИИ

## procedure SET\_LOCATOR\_MODE

(WS : in WS\_ID;  
 DEVICE : in LOCATOR\_DEVICE\_NUMBER;  
 MODE : in OPERATING\_MODE;  
 SWITCH : in ECHO\_SWITCH);

## SET STROKE MODE

Уровень 0b

## ЗАДАТЬ РЕЖИМ УСТРОЙСТВА ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИИ

## procedure SET\_STROKE\_MODE

(WS	: in WS_ID;
DEVICE	: in STROKE_DEVICE_
	NUMBER;
MODE	: in OPERATION_MODE;
SWITCH	: in ECHO_SWITCH);

SET VALUATOR MODE Уровень 0b  
 ЗАДАТЬ РЕЖИМ УСТРОЙСТВА ЧИСЛА

```
procedure SET_VALUATOR_MODE
  (WS : in WS_ID;
   DEVICE : in VALUATOR_DEVICE_
   NUMBER;
   MODE : in OPERATING_MODE;
   SWITCH : in ECHO_SWITCH);
```

SET CHOICE MODE Уровень 0b  
 ЗАДАТЬ РЕЖИМ УСТРОЙСТВА ВЫБОРА

```
procedure SET_CHOICE_MODE
  (WS : in WS_ID;
   DEVICE : in CHOICE_DEVICE_
   NUMBER;
   MODE : in OPERATING_MODE;
   SWITCH : in ECHO_SWITCH);
```

SET PICK MODE Уровень 1b  
 ЗАДАТЬ РЕЖИМ УСТРОЙСТВА УКАЗАНИЯ

```
procedure SET_PICK_MODE
  (WS : in WS_ID;
   DEVICE : in PICK_DEVICE_NUMBER;
   MODE : in OPERATING_MODE;
   SWITCH : in ECHO_SWITCH);
```

SET STRING MODE Уровень 0b  
 ЗАДАТЬ РЕЖИМ УСТРОЙСТВА ВВОДА СТРОКИ

```
procedure SET_STRING_MODE
  (WS : in WS_ID;
   DEVICE : in STRING_DEVICE_
   NUMBER;
   MODE : in OPERATING_MODE;
   SWITCH : in ECHO_SWITCH);
```

REQUEST LOCATOR Уровень 0b  
 ЗАПРОСИТЬ ВВОД ПОЗИЦИИ

```
procedure REQUEST_LOCATOR
```

(WS	: in WS_ID;
DEVICE	: in LOCATOR_DEVICE_
	NUMBER;
STATUS	: out INPUT_STATUS;
TRANSFORMATION	: out TRANSFORMATION_
	NUMBER;
POSITION	: out WC POINT);

REQUEST STROKE Уровень 0b  
 ЗАПРОСИТЬ ВВОД ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИИ  
 procedure REQUEST\_STROKE

(WS	: in WS_ID;
DEVICE	: in STROKE_DEVICE_
	NUMBER;
STATUS	: out INPUT_STATUS;
TRANSFORMATION	: out TRANSFORMATION_
	NUMBER;
STROKE_POINTS	: out WC. POINT_LIST);

REQUEST VALUATOR Уровень 0b  
 ЗАПРОСИТЬ ВВОД ЧИСЛА  
 procedure REQUEST\_VALUATOR

(WS	: in WS_ID;
DEVICE	: in VALUATOR_DEVICE_
	NUMBER;
STATUS	: out INPUT_STATUS;
VALUE	: out VALUATOR_INPUT_
	VALUE);

REQUEST CHOICE Уровень 0b  
 ЗАПРОСИТЬ УСТРОЙСТВО ВЫБОРА  
 procedure REQUEST\_CHOICE

(WS	: in WS_ID;
DEVICE	: in CHOICE_DEVICE_
	NUMBER;
STATUS	: out CHOICE_REQUEST_
	STATUS;
CHOICE_NUMBER	: out CHOICE_VALUE);

REQUEST PICK Уровень 1b  
 ЗАПРОСИТЬ УКАЗАНИЯ  
 procedure REQUEST\_PICK

(WS	: in WS_ID;
-----	-------------

DEVICE	: in PICK_DEVICE_NUMBER;
STATUS	: out PICK_REQUEST_STATUS;
SEGMENT	: out SEGMENT_NAME;
PICK	: out PICK_ID);

REQUEST STRING Уровень 0b

ЗАПРОСИТЬ ВВОД СТРОКИ

procedure REQUEST\_STRING

(WS	: in WS_ID;
DEVICE	: in STRING_DEVICE_NUMBER;
STATUS	: out INPUT_STATUS;
CHAR_STRING	: out INPUT_STRING);

SAMPLE LOCATOR Уровень 0c

ОПРОСИТЬ ВВОД ПОЗИЦИИ

procedure SAMPLE\_LOCATOR

(WS	: in WS_ID;
DEVICE	: in LOCATOR_DEVICE_NUMBER;
TRANSFORMATION	: out TRANSFORMATION_NUMBER;
POSITION	: out WC_POINT);

SAMPLE STROKE Уровень 0c

ОПРОСИТЬ ВВОД ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИИ

procedure SAMPLE\_STROKE

(WS	: in WS_ID;
DEVICE	: in STROKE_DEVICE_NUMBER;
TRANSFORMATION	: out TRANSFORMATION_NUMBER;
STROKE_POINTS	: out WC_POINT_LIST);

SAMPLE VALUATOR Уровень 0c

ОПРОСИТЬ ВВОД ЧИСЛА

procedure SAMPLE\_VALUATOR

(WS	: in WS_ID;
DEVICE	: in VALUATOR_DEVICE_NUMBER;
VALUE	: out VALUATOR_INPUT_VALUE);



GET LOCATOR ПОЛУЧИТЬ ПОЗИЦИЮ procedure GET_LOCATOR (TRANSFORMATION POSITION	: out TRANSFORMATION_ NUMBER; : out WC. POINT);	Уровень 0c
GET STROKE ПОЛУЧИТЬ ПОСЛЕДОВАТЕЛЬНОСТЬ ПОЗИЦИИ procedure GET_STROKE (TRANSFORMATION STROKE_POINTS	: out TRANSFORMATION_ NUMBER; : out WC. POINT_LIST);	Уровень 0c
GET VALUATOR ПОЛУЧИТЬ ЧИСЛО procedure GET_VALUATOR (VALUE	: out VALUATOR_INPUT_ VALUE);	Уровень 0c
GET CHOICE ПОЛУЧИТЬ АЛЬТЕРНАТИВУ procedure GET_CHOICE (STATUS CHOICE_NUMBER	: out CHOICE_STATUS; : out CHOICE_VALUE);	Уровень 0c
GET PICK ПОЛУЧИТЬ УКАЗАТЕЛЬ procedure GET_PICK (STATUS SEGMENT PICK	: out PICK_STATUS; : out SEGMENT_NAME; : out PICK_ID);	Уровень 1c
GET STRING ПОЛУЧИТЬ СТРОКУ procedure GET_STRING (CHAR_STRING	: out INPUT_STRING);	Уровень 0c
WRITE ITEM TO GKSM ЗАПИСЬ В МЕТАФАЙЛ procedure WRITE_ITEM_TO_GKSM (WS ITEM	: in WS_ID; : in GKSM_DATA_RECORD);	Уровень 0a

GET ITEM TYPE FROM GKSM ПОЛУЧИТЬ ТИП ЗАПИСИ ИЗ ЯГС	Уровень 0a
procedure GET_ITEM_TYPE_FROM_GKSM (WS : in WS_ID; TYPE_OF_ITEM : out GKSM_ITEM_TYPE; LENGTH : out NATURAL);	
READ ITEM FROM GKSM ПРОЧИТАТЬ ЗАПИСЬ ИЗ ЯГС	Уровень 0a
procedure READ_ITEM_FROM_GKSM (WS : in WS_ID; MAX_LENGTH : in NATURAL; ITEM : out GKSM_DATA_RECORD);	
INTERPRET ITEM ИНТЕРПРЕТИРОВАТЬ ЗАПИСЬ	Уровень 0a
procedure INTERPRET_ITEM (ITEM : in GKSM_DATA_RECORD);	
INQUIRE OPERATING STATE VALUE УЗНАТЬ ФУНКЦИОНАЛЬНОЕ СОСТОЯНИЕ	Уровень 0a
procedure INQ_OPERATING_STATE_VALUE (VALUE : out OPERATING_STATE);	
INQUIRE LEVEL OF GKS УЗНАТЬ УРОВЕНЬ ЯГС	Уровень 0a
procedure INQ_LEVEL_OF_GKS (ERROR_INDICATOR : out ERROR_NUMBER; LEVEL : out GSK_LEVEL);	
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES УЗНАТЬ ДОСТУПНЫЕ ТИПЫ СТАНЦИЙ	Уровень 0a
procedure INQ_LIST_OF_AVAILABLE_WS_TYPES (ERROR_INDICATOR : out ERROR_NUMBER; TYPES : out WS_TYPES_LIST_OF);	
INQUIRE WORKSTATION MAXIMUM NUMBERS УЗНАТЬ ДОПУСТИМЫЕ КОЛИЧЕСТВА СТАНЦИЙ	Уровень 1a
procedure INQ_WS_MAX_NUMBERS (ERROR_INDICATOR : out ERROR_NUMBER; MAX_OPEN_WS : out POSITIVE);	

MAX\_ACTIVE\_WS : out POSITIVE;  
 MAX\_SEGMENT\_WS : out POSITIVE;

INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER Уровень 0a

УЗНАТЬ МАКСИМАЛЬНЫЙ НОМЕР ПРЕОБРАЗОВАНИЯ  
 НОРМИРОВАНИЯ

procedure INQ\_MAX\_NORMALIZATION\_TRANSFORMATION\_NUMBER  
 (ERROR\_INDICATOR : out ERROR\_NUMBER;  
 TRANSFORMATION : out TRANSFORMATION\_NUMBER);

INQUIRE SET OF OPEN WORKSTATIONS Уровень 0a

УЗНАТЬ НАБОР ОТКРЫТЫХ СТАНЦИЙ

procedure INQ\_SET\_OF\_OPEN\_WS  
 (ERROR\_INDICATOR : out ERROR\_NUMBER;  
 WS : out WS\_IDS\_LIST\_OF);

INQUIRE SET OF ACTIVE WORKSTATIONS Уровень 1a

УЗНАТЬ НАБОР АКТИВНЫХ СТАНЦИЙ

procedure INQ\_SET\_OF\_ACTIVE\_WS  
 (ERROR\_INDICATOR : out ERROR\_NUMBER;  
 WS : out WS\_IDS\_LIST\_OF);

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES Уровень 0a

УЗНАТЬ ТЕКУЩИЕ ЗНАЧЕНИЯ АТТРИБУТОВ ПРИМИТИВОВ

procedure INQ\_CURRENT\_PRIMITIVE\_ATTRIBUTE\_VALUES  
 (ERROR\_INDICATOR : out ERROR\_NUMBER;  
 ATTRIBUTES : out PRIMITIVE\_ATTRIBUTE\_VALUES);

procedure INQ\_POLYLINE\_INDEX  
 (ERROR\_INDICATOR : out ERROR\_NUMBER;  
 INDEX : out POLYLINE\_INDEX);

procedure INQ\_POLYMARKER\_INDEX  
 (ERROR\_INDICATOR : out ERROR\_NUMBER;  
 INDEX : out POLYMARKER\_INDEX);

procedure INQ\_TEXT\_INDEX  
 (ERROR\_INDICATOR : out ERROR\_NUMBER;  
 INDEX : out TEXT\_INDEX);

```

: procedure INQ_CHAR_HEIGHT
: (ERROR_INDICATOR : out ERROR_NUMBER;
: HEIGHT : out WC_MAGNITUDE);
: procedure INQ_CHAR_UP_VECTOR
: (ERROR_INDICATOR : out ERROR_NUMBER;
: VECTOR : out WC_VECTOR);
: procedure INQ_CHAR_WIDTH
: (ERROR_INDICATOR : out ERROR_NUMBER;
: WIDTH : out WC_MAGNITUDE);
: procedure INQ_CHAR_BASE_VECTOR
: (ERROR_INDICATOR : out ERROR_NUMBER;
: VECTOR : out WC_VECTOR);
: procedure INQ_TEXT_PATH
: (ERROR_INDICATOR : out ERROR_NUMBER;
: PATH : out TEXT_PATH);
: procedure INQ_TEXT_ALIGNMENT
: (ERROR_INDICATOR : out ERROR_NUMBER;
: ALIGNMENT : out TEXT_ALIGNMENT);
: procedure INQ_FILL_AREA_INDEX
: (ERROR_INDICATOR : out ERROR_NUMBER;
: INDEX : out FILL_AREA_INDEX);
: procedure INQ_PATTERN_WIDTH_VECTOR
: (ERROR_INDICATOR : out ERROR_NUMBER;
: WIDTH : out WC_VECTOR);
: procedure INQ_PATTERN_HEIGHT_VECTOR
: (ERROR_INDICATOR : out ERROR_NUMBER;
: VECTOR : out WC_VECTOR);
: procedure INQ_PATTERN_REFERENCE_POINT
: (ERROR_INDICATOR : out ERROR_NUMBER;
: REFERENCE_POINT : out WC_POINT);

```

---

INQUIRE CURRENT PICK IDENTIFIER Уровень 1b  
 VALUE

УЗНАТЬ ЗНАЧЕНИЕ ИДЕНТИФИКАТОРА УКАЗАНИЯ

```

: procedure INQ_CURRENT_PICK_ID_VALUE
: (ERROR_INDICATOR : out ERROR_NUMBER;
: PICK : out PICK_ID);

```

---

INQUIRE CURRENT INDIVIDUAL Уровень 0a  
 ATTRIBUTE VALUES

УЗНАТЬ ТЕКУЩИЕ ЗНАЧЕНИЯ ИНДИВИДУАЛЬНЫХ  
 АТТРИБУТОВ

```

procedure INQ_CURRENT_INDIVIDUAL_ATTRIBUTE_
                                _VALUES
(ERROR_INDICATOR      : out ERROR_NUMBER;
ATTRIBUTES            : out INDIVIDUAL_ATTRIBUTE
                                _VALUES);

procedure INQ_LINETYPE
(ERROR_INDICATOR      : out ERROR_NUMBER;
TYPE_OF_LINE         : out LINETYPE);

procedure INQ_LINEWIDTH_SCALE_FACTOR
(ERROR_INDICATOR      : out ERROR_NUMBER;
WIDTH                : out LINEWIDTH);

procedure INQ_POLYLINE_COLOUR_INDEX
(ERROR_INDICATOR      : out ERROR_NUMBER;
LINE_COLOUR          : out COLOUR_INDEX);

procedure INQ_POLYMARKER_TYPE
(ERROR_INDICATOR      : out ERROR_NUMBER;
TYPE_OF_MARKER       : out MARKER_TYPE);

procedure INQ_POLYMARKER_SIZE_SCALE_FACTOR
(ERROR_INDICATOR      : out ERROR_NUMBER;
SIZE                 : out MARKER_SIZE);

procedure INQ_POLYMARKER_COLOUR_INDEX
(ERROR_INDICATOR      : out ERROR_NUMBER;
MARKER_COLOUR        : out COLOUR_INDEX);

procedure INQ_TEXT_FONT_AND_PRECISION
(ERROR_INDICATOR      : out ERROR_NUMBER;
FONT_PRECISION        : out TEXT_FONT_PRECISION);

procedure INQ_CHAR_EXPANSION_FACTOR
(ERROR_INDICATOR      : out ERROR_NUMBER;
EXPANSION             : out CHAR_EXPANSION);

procedure INQ_TEXT_COLOUR_INDEX
(ERROR_INDICATOR      : out ERROR_NUMBER;
TEXT_COLOUR           : out COLOUR_INDEX);

procedure INQ_FILL_AREA_INTERIOR_STYLE
(ERROR_INDICATOR      : out ERROR_NUMBER;
INTERIOR              : out INTERIOR_STYLE);

procedure INQ_FILL_AREA_STYLE_INDEX
(ERROR_INDICATOR      : out ERROR_NUMBER;
STYLE                 : out STYLE_INDEX);

procedure INQ_FILL_AREA_COLOUR_INDEX
(ERROR_INDICATOR      : out ERROR_NUMBER;
FILL_AREA_COLOUR      : out COLOUR_INDEX);

```

procedure INQ_LIST_OF_ASF (ERROR_INDICATOR : out ERROR_NUMBER; LIST : out ASF_LIST);	
INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER УЗНАТЬ НОМЕР ТЕКУЩЕГО ПРЕОБРАЗОВАНИЯ НОРМИРОВАНИЯ	Уровень 0a
procedure INQ_CURRENT_NORMALIZATION_ _TRANSFORMATION_NUMBER (ERROR_INDICATOR : out ERROR_NUMBER; TRANSFORMATION : out TRANSFORMATION_ _NUMBER);	
INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS УЗНАТЬ СПИСОК ПРЕОБРАЗОВАНИЙ НОРМИРОВАНИЯ	Уровень 0a
procedure INQ_LIST_OF_NORMALIZATION_ _TRANSFORMATION_NUMBERS (ERROR_INDICATOR : out ERROR_NUMBER; LIST : out TRANSFORMATION_ _PRIORITY_LIST);	
INQUIRE NORMALIZATION TRANSFORMATION УЗНАТЬ ПРЕОБРАЗОВАНИЯ НОРМИРОВАНИЯ	Уровень 0a
procedure INQ_NORMALIZATION_TRANSFORMATION (TRANSFORMATION : in TRANSFORMATION_ _NUMBER; ERROR_INDICATOR : out ERROR_NUMBER; WINDOW_LIMITS : out WC_RECTANGLE_LIMITS; VIEWPORT_LIMITS : out NDC_RECTANGLE_ _LIMITS);	
INQUIRE CLIPPING УЗНАТЬ ЗНАЧЕНИЯ ОТСЕЧЕНИЯ	Уровень 0a
procedure INQ_CLIPPING (ERROR_INDICATOR : out ERROR_NUMBER; CLIPPING : out CLIPPING_INDICATOR; CLIPPING_RECTANGLE : out NDC_RECTANGLE_ _LIMITS);	
INQUIRE NAME OF OPEN SEGMENT УЗНАТЬ ИМЯ ОТКРЫТОГО СЕГМЕНТА	Уровень 1a

```

procedure INQ_NAME_OF_OPEN_SEGMENT
(ERROR_INDICATOR      : out ERROR_NUMBER;
SEGMENT               : out SEGMENT_NAME);

```

INQUIRE SET OF SEGMENT NAMES IN USE Уровень 1a

УЗНАТЬ ИМЕНА СУЩЕСТВУЮЩИХ СЕГМЕНТОВ

```

procedure INQ_SET_OF_SEGMENT_NAMES_IN_USE
(ERROR_INDICATOR      : out ERROR_NUMBER;
SEGMENTS              : out SEGMENT_NAMES_LIST_OF);

```

INQUIRE MORE SIMULTANEOUS EVENTS Уровень 0c

УЗНАТЬ НАЛИЧИЕ ОДНОВРЕМЕННЫХ СОБЫТИЙ

```

procedure INQ_MORE_SIMULTANEOUS_EVENTS
(ERROR_INDICATOR      : out ERROR_NUMBER;
EVENTS                : out MORE_EVENTS);

```

INQUIRE WORKSTATION CONNECTION AND TYPE Уровень 0a

УЗНАТЬ ТИП И ИДЕНТИФИКАТОР СВЯЗИ СТАНЦИИ

```

procedure INQ_WS_CONNECTION_AND_TYPE
(WS                   : in WS_ID;
ERROR_INDICATOR      : out ERROR_NUMBER;
CONNECTION            : out VARIABLE_CONNECTION_ID;
TYPE_OF_WS           : out WS_TYPE);

```

INQUIRE WORKSTATION STATE Уровень 0a

УЗНАТЬ СОСТОЯНИЕ СТАНЦИИ

```

procedure INQ_WS_STATE
(WS                   : in WS_ID;
ERROR_INDICATOR      : out ERROR_NUMBER;
STATE                 : out WS_STATE);

```

INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES Уровень 0a

УЗНАТЬ РЕЖИМЫ ЗАДЕРЖКИ И ОБНОВЛЕНИЯ СТАНЦИИ

```

procedure INQ_WS_DEFERRAL_AND_UPDATE_STATES
(WS                   : in WS_ID;
ERROR_INDICATOR      : out ERROR_NUMBER;

```

DEFERRAL : out DEFERRAL\_MODE;  
 REGENERATION : out REGENERATION\_MODE;  
 DISPLAY : out DISPLAY\_SURFACE\_...  
 \_EMPTY;  
 FRAME\_ACTION : out NEW\_FRAME...  
 \_NECESSARY);

INQUIRE LIST OF POLYLINE INDICES Уровень 1a  
 УЗНАТЬ ИНДЕКСЫ ЛОМАНОЙ  
 procedure INQ\_LIST\_OF\_POLYLINE\_INDICES  
 (WS : in WS\_ID;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 INDICES : out POLYLINE\_INDICES.  
 LIST-OF);

INQUIRE POLYLINE REPRESENTATION Уровень 1a  
 УЗНАТЬ ПРЕДСТАВЛЕНИЕ ЛОМАНОЙ  
 procedure INQ\_POLYLINE\_REPRESENTATION  
 (WS : in WS\_ID;  
 INDEX : in POLYLINE\_INDEX;  
 RETURNED\_VALUES : in RETURN\_VALUE\_TYPE;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 TYPE\_OF\_LINE : out LINETYPE;  
 WIDTH : out LINEWIDTH;  
 LINE\_COLOUR : out COLOUR\_INDEX);

INQUIRE LIST OF POLYMARKER INDICES Уровень 1a  
 УЗНАТЬ ИНДЕКСЫ ПОЛИМАРКЕРА  
 procedure INQ\_LIST\_OF\_POLYMARKER\_INDICES  
 (WS : in WS\_ID;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 INDICES : out POLYMARKER\_INDICES.  
 LIST-OF);

INQUIRE POLYMARKER REPRESENTATION Уровень 1a  
 УЗНАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИМАРКЕРА  
 procedure INQ\_POLYMARKER\_REPRESENTATION  
 (WS : in WS\_ID;  
 INDEX : in POLYMARKER\_INDEX;  
 RETURNED\_VALUES : in RETURN\_VALUE\_TYPE;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;

TYPE\_OF\_MARKER : out MARKER\_TYPE;  
 SIZE : out MARKER\_SIZE;  
 MARKER\_COLOUR : out COLOUR\_INDEX);

INQUIRE LIST OF TEXT INDICES Уровень 1a  
 УЗНАТЬ ИНДЕКСЫ ТЕКСТА

procedure INQ\_LIST\_OF\_TEXT\_INDICES  
 (WS : in WS\_ID;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 INDICES : out TEXT\_INDICES\_LIST\_OF);

INQUIRE TEXT REPRESENTATION Уровень 1a  
 УЗНАТЬ ПРЕДСТАВЛЕНИЕ ТЕКСТА

procedure INQ\_TEXT\_REPRESENTATION  
 (WS : in WS\_ID;  
 INDEX : in TEXT\_INDEX;  
 RETURNED\_VALUES : in RETURN\_VALUE\_TYPE;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 FONT\_PRECISION : out TEXT\_FONT\_PRECISION;  
 EXPANSION : out CHAR\_EXPANSION;  
 SPACING : out CHAR\_SPACING;  
 TEXT\_COLOUR : out COLOUR\_INDEX);

INQUIRE TEXT EXTENT Уровень 0a  
 УЗНАТЬ ГАБАРИТЫ ТЕКСТА

procedure INQ\_TEXT\_EXTENT  
 (WS : in WS\_ID;  
 POSITION : in WC\_POINT;  
 CHAR\_STRING : in STRING;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 CONCATENATION\_POINT : out WC\_POINT;  
 TEXT\_EXTENT : out TEXT\_EXTENT\_PARALLELOGRAM);

INQUIRE LIST OF FILL AREA INDICES Уровень 1a  
 УЗНАТЬ ИНДЕКСЫ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ

procedure INQ\_LIST\_OF\_FILL\_AREA\_INDICES  
 (WS : in WS\_ID;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 INDICES : out FILL\_AREA\_INDICES\_LIST\_OF);

---

INQUIRE FILL AREA REPRESENTATION Уровень 1a  
 УЗНАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИГОНАЛЬНОЙ  
 ОБЛАСТИ

```

procedure INQ_FILL_AREA_REPRESENTATION
  (WS                : in WS_ID;
   INDEX             : in FILL_AREA_INDEX;
   RETURNED_VALUES  : in RETURN_VALUE_TYPE;
   ERROR_INDICATOR  : out ERROR_NUMBER;
   INTERIOR          : out INTERIOR_STYLE;
   STYLE             : out STYLE_INDEX;
   FILL_AREA_COLOUR : out COLOUR_INDEX);
  
```

---

INQUIRE LIST OF PATTERN INDICES Уровень 1a  
 УЗНАТЬ ИНДЕКСЫ ШАБЛОНА

```

procedure INQ_LIST_OF_PATTERN_INDICES
  (WS                : in WS_ID;
   ERROR_INDICATOR  : out ERROR_NUMBER;
   INDICES           : out PATTERN_INDICES.
                    LIST_OF);
  
```

---

INQUIRE PATTERN REPRESENTATION Уровень 1a  
 УЗНАТЬ ПРЕДСТАВЛЕНИЕ ШАБЛОНА

```

procedure INQ_PATTERN_REPRESENTATION
  (WS                : in WS_ID;
   INDEX             : in PATTERN_INDEX;
   RETURNED_VALUES  : in RETURN_VALUE_TYPE;
   ERROR_INDICATOR  : out ERROR_NUMBER;
   PATTERN          : out VARIABLE_COLOUR_
                    _MATRIX);
  
```

---

INQUIRE LIST OF COLOUR INDICES Уровень 0a  
 УЗНАТЬ ИНДЕКСЫ ЦВЕТА

```

procedure INQ_LIST_OF_COLOUR_INDICES
  (WS                : in WS_ID;
   ERROR_INDICATOR  : out ERROR_NUMBER;
   INDICES           : out COLOUR_INDICES.
                    LIST_OF);
  
```

---

INQUIRE COLOUR REPRESENTATION Уровень 0a  
 УЗНАТЬ ПРЕДСТАВЛЕНИЕ ЦВЕТА

```

procedure INQ_COLOUR_REPRESENTATION
  (WS                : in WS_ID;
   INDEX             : in COLOUR_INDEX;
  
```

RETURNED\_VALUES : in RETURN\_VALUE\_TYPE;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 RGB\_COLOUR : out COLOUR\_  
 \_REPRESENTATION);

INQUIRE WORKSTATION Уровень 0a  
 TRANSFORMATION

УЗНАТЬ ПРЕОБРАЗОВАНИЕ СТАНЦИИ

procedure INQ\_WS\_TRANSFORMATION

(WS : in WS\_ID;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 UPDATE : out UPDATE\_STATE;  
 REQUESTED\_WINDOW : out NDC\_RECTANGLE\_  
 \_LIMITS;  
 CURRENT\_WINDOW : out NDC\_RECTANGLE\_  
 \_LIMITS;  
 REQUESTED\_VIEWPORT : out DC\_RECTANGLE\_LIMITS;  
 CURRENT\_VIEWPORT : out DC\_RECTANGLE\_  
 \_LIMITS);

INQUIRE SET OF SEGMENT NAMES ON Уровень 1a  
 WORKSTATION

УЗНАТЬ ИМЕНА СЕГМЕНТОВ, ХРАНИМЫХ  
 НА СТАНЦИИ

procedure INQ\_SET\_OF\_SEGMENT\_NAMES\_ON\_WS

(WS : in WS\_ID;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 SEGMENTS : out SEGMENT\_NAMES\_  
 LIST\_OF);

INQUIRE LOCATOR DEVICE STATE Уровень 0b  
 УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА ВВОДА ПОЗИЦИИ

procedure INQ\_LOCATOR\_DEVICE\_STATE

(WS : in WS\_ID;  
 DEVICE : in LOCATOR\_DEVICE\_  
 \_NUMBER;  
 RETURNED\_VALUES : in RETURN\_VALUE\_TYPE;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 MODE : out OPERATING\_MODE;  
 SWITCH : out ECHO\_SWITCH;  
 INITIAL\_TRANSFOR- : out TRANSFORMATION\_  
 MATION \_NUMBER;  
 INITIAL\_POSITION : out WC\_POINT;

ECHO\_AREA : out DC.RECTANGLE\_ \_LIMITS;  
 DATA\_RECORD : out LOCATOR\_DATA\_ \_RECORD);

INQUIRE STROKE DEVICE STATE Уровень 0b

УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА ВВОДА  
 ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИИ

procedure INQ\_STROKE\_DEVICE\_STATE  
 (WS : in WS\_ID;  
 DEVICE : in STROKE\_DEVICE\_ \_NUMBER;  
 RETURNED\_VALUES : in RETURN\_VALUE\_TYPE;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 MODE : out OPERATING\_MODE;  
 SWITCH : out ECHO\_SWITCH;  
 INITIAL\_TRANSFORMATION : out TRANSFORMATION\_ \_NUMBER;  
 INITIAL\_STROKE\_POINTS : out WC.POINT\_LIST;  
 ECHO\_AREA : out DC.RECTANGLE\_LIMITS;  
 DATA\_RECORD : out STROKE\_DATA\_RECORD);

INQUIRE VALUATOR DEVICE STATE Уровень 0b

УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА ВВОДА ЧИСЛА

procedure INQ\_VALUATOR\_DEVICE\_STATE  
 (WS : in WS\_ID;  
 DEVICE : in VALUATOR\_DEVICE\_ \_NUMBER;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 MODE : out OPERATING\_MODE;  
 SWITCH : out ECHO\_SWITCH;  
 INITIAL\_VALUE : out VALUATOR\_INPUT\_ \_VALUE;  
 ECHO\_AREA : out DC.RECTANGLE\_LIMITS;  
 DATA\_RECORD : out VALUATOR\_DATA\_ \_RECORD);

INQUIRE CHOICE DEVICE STATE Уровень 0b

УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА ВЫБОРА

procedure INQ\_CHOICE\_DEVICE\_STATE  
 (WS : in WS\_ID;  
 DEVICE : in CHOICE\_DEVICE\_ \_NUMBER;



---

INQUIRE WORKSTATION \_CLASSIFICATION Уровень 0a

УЗНАТЬ КЛАСС СТАНЦИИ

```

procedure INQ_WS_CLASSIFICATION
  (TYPE_OF_WS           : in WS_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   CLASS                : out DISPLAY_CLASS);
  
```

---

INQUIRE DISPLAY SPACE SIZE Уровень 0a

УЗНАТЬ РАЗМЕР НОСИТЕЛЯ ИЗОБРАЖЕНИЯ

```

procedure INQ_DISPLAY_SPACE_SIZE
  (TYPE_OF_WS           : in WS_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   UNITS                : out DC_UNITS;
   MAX_DC_SIZE         : out DC_SIZE;
   MAX_RASTER_UNIT_SIZE : out RASTER_UNIT_SIZE);
  
```

---

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION\_ATTRIBUTES Уровень 1a

УЗНАТЬ СПОСОБ ДИНАМИЧЕСКОГО ОБНОВЛЕНИЯ ХАРАКТЕРИСТИК ИЗОБРАЖЕНИЯ НА СТАНЦИИ

```

procedure INQ_DYNAMIC_MODIFICATION_OF_WS_ATTRIBUTES
  (TYPE_OF_WS           : in WS_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   POLYLINE_REPRESENTATION : out DYNAMIC_MODIFICATION;
   POLYMARKER_REPRESENTATION : out DYNAMIC_MODIFICATION;
   TEXT_REPRESENTATION : out DYNAMIC_MODIFICATION;
   FILL_AREA_REPRESENTATION : out DYNAMIC_MODIFICATION;
   PATTERN_REPRESENTATION : out DYNAMIC_MODIFICATION;
   COLOUR_REPRESENTATION : out DYNAMIC_MODIFICATION;
   TRANSFORMATION      : out DYNAMIC_MODIFICATION);
  
```

---

INQUIRE DEFAULT DEFERRAL STATE Уровень 1a  
VALUES

УЗНАТЬ РЕЖИМ ЗАДЕРЖКИ ПО УМОЛЧАНИЮ

procedure INQ\_DEFAULT\_DEFERRAL\_STATE\_VALUES

(TYPE\_OF\_WS : in WS\_TYPE;  
ERROR\_INDICATOR : out ERROR\_NUMBER;  
DEFERRAL : out DEFERRAL\_MODE;  
REGENERATION : out REGENERATION\_MODE);

---

INQUIRE POLYLINE FACILITIES Уровень 0a

УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ ЛОМАННОЙ

procedure INQ\_POLYLINE\_FACILITIES

(TYPE\_OF\_WS : in WS\_TYPE;  
ERROR\_INDICATOR : out ERROR\_NUMBER;  
LIST\_OF\_TYPES : out LINETYPES.LIST\_OF;  
NUMBER\_OF\_WIDTHS : out NATURAL;  
NOMINAL\_WIDTH : out DC.MAGNITUDE;  
RANGE\_OF\_WIDTHS : out DC.RANGE\_OF\_  
\_MAGNITUDES;  
NUMBER\_OF\_INDICES : out NATURAL);

---

INQUIRE PREDEFINED POLYLINE Уровень 0a

REPRESENTATION

УЗНАТЬ ПРЕДСТАВЛЕНИЕ ЛОМАННОЙ ПО УМОЛЧАНИЮ

procedure INQ\_PREDEFINED\_POLYLINE\_REPRESENTATION

(TYPE\_OF\_WS : in WS\_TYPE;  
INDEX : in POLYLINE\_INDEX;  
ERROR\_INDICATOR : out ERROR\_NUMBER;  
TYPE\_OF\_LINE : out LINETYPE;  
WIDTH : out LINEWIDTH;  
LINE\_COLOUR : out COLOUR\_INDEX);

---

INQUIRE POLYMARKER FACILITIES Уровень 0a

УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ

ПОЛИМАРКЕРА

procedure INQ\_POLYMARKER\_FACILITIES

(TYPE\_OF\_WS : in WS\_TYPE;  
ERROR\_INDICATOR : out ERROR\_NUMBER;  
LIST\_OF\_TYPES : out MARKER\_TYPES.LIST\_OF;  
NUMBER\_OF\_SIZES : out NATURAL;  
NOMINAL\_SIZE : out DC.MAGNITUDE;

RANGE\_OF\_SIZES : out DC.RANGE\_OF\_  
 \_MAGNITUDES;  
 NUMBER\_OF\_INDICES : out NATURAL);

INQUIRE PREDEFINED POLYMARKER Уровень 0a

REPRESENTATION  
 УЗНАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИМАРКЕРА  
 ПО УМОЛЧАНИЮ

procedure INQ\_PREDEFINED\_POLYMARKER\_  
 \_REPRESENTATION

(TYPE\_OF\_WS : in WS\_TYPE;  
 INDEX : in POLYMARKER\_INDEX;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 TYPE\_OF\_MARKER : out MARKER\_TYPE;  
 SIZE : out MARKER\_SIZE;  
 MARKER\_COLOUR : out COLOUR\_INDEX);

INQUIRE TEXT FACILITIES Уровень 0a

УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ ТЕКСТА

procedure INQ\_TEXT\_FACILITIES

(TYPE\_OF\_WS : in WS\_TYPE;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 LIST\_OF\_FONT\_  
 \_PRECISION\_PAIRS : out TEX\_FONT\_PRECISIONS.  
 LIST\_OF;  
 NUMBER\_OF\_HEIGHTS : out NATURAL;  
 RANGE\_OF\_HEIGHTS : out DC.RANGE\_OF\_  
 \_MAGNITUDES;  
 NUMBER\_OF\_  
 \_EXPANSIONS : out NATURAL;  
 EXPANSION\_RANGE : out RANGE\_OF\_EXPANSIONS;  
 NUMBER\_OF\_INDICES : out NATURAL);

INQUIRE PREDEFINED TEXT Уровень 0a

REPRESENTATION  
 УЗНАТЬ ПРЕДСТАВЛЕНИЕ ТЕКСТА ПО УМОЛЧАНИЮ

procedure INQ\_PREDEFINED\_TEXT\_REPRESENTATION

(TYPE\_OF\_WS : in WS\_TYPE;  
 INDEX : in TEXT\_INDEX;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 FONT\_PRECISION : out TEXT\_FONT\_PRECISION;  
 EXPANSION : out CHAR\_EXPANSION;  
 SPACING : out CHAR\_SPACING;  
 TEXT\_COLOUR : out COLOUR\_INDEX);

---

INQUIRE FILL AREA FACILITIES Уровень 0a  
 УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ  
 ПОЛИГОНАЛЬНОЙ ОБЛАСТИ.

```

procedure INQ_FILL_AREA_FACILITIES
  (TYPE_OF_WS           : in WS_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   LIST_OF_INTERIOR_   : out INTERIOR_STYLES;
   _STYLES              LIST_OF;
   LIST_OF_HATCH_     : out HATCH_STYLES.LIST_OF;
   _STYLES
   NUMBER_OF_INDICES   : out NATURAL);
  
```

---

INQUIRE PREDEFINED FILL AREA Уровень 0a  
 REPRESENTATION  
 УЗНАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ  
 ПО УМОЛЧАНИЮ

```

procedure INQ_PREDEFINED_FILL_AREA_
  _REPRESENTATION
  (TYPE_OF_WS           : in WS_TYPE;
   INDEX                : in FILL_AREA_INDEX;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   INTERIOR             : out INTERIOR_STYLE;
   STYLE                : out STYLE_INDEX;
   FILL_AREA_COLOUR    : out COLOUR_INDEX);
  
```

---

INQUIRE PATTERN FACILITIES Уровень 0a  
 УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ ШАБЛОНА

```

procedure INQ_PATTERN_FACILITIES
  (TYPE_OF_WS           : in WS_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   NUMBER_OF_INDICES   : out NATURAL);
  
```

---

INQUIRE PREDEFINED PATTERN Уровень 0a  
 REPRESENTATION  
 УЗНАТЬ ПРЕДСТАВЛЕНИЕ ШАБЛОНА ПО УМОЛЧАНИЮ

```

procedure INQ_PREDEFINED_PATTERN_REPRESENTATION
  (TYPE_OF_WS           : in WS_TYPE;
   INDEX                : in PATTERN_INDEX;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   PATTERN              : out VARIABLE_COLOUR_
   _MATRIX);
  
```

---

INQUIRE COLOUR FACILITIES Уровень 0a  
 УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ ЦВЕТА

procedure INQ\_COLOUR\_FACILITIES  
 (TYPE\_OF\_WS : in WS\_TYPE;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 NUMBER\_OF\_COLOURS : out NATURAL;  
 AVAILABLE\_COLOUR : out COLOUR\_AVAILABLE;  
 NUMBER\_OF\_COLOUR\_INDICES : out NATURAL);

---

INQUIRE PREDEFINED COLOUR REPRESENTATION Уровень 0a  
 УЗНАТЬ ПРЕДСТАВЛЕНИЕ ЦВЕТА ПО УМОЛЧАНИЮ

procedure INQ\_PREDEFINED\_COLOUR\_REPRESENTATION  
 (TYPE\_OF\_WS : in WS\_TYPE;  
 INDEX : in COLOUR\_INDEX;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 RGB\_COLOUR : out COLOUR\_REPRESENTATION);

---

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES Уровень 0a  
 УЗНАТЬ ИДЕНТИФИКАТОРЫ ДОСТУПНЫХ  
 ОБОБЩЕННЫХ ПРИМИТИВОВ ВЫВОДА

procedure INQ\_LIST\_OF\_AVAILABLE\_GDP  
 (TYPE\_OF\_WS : in WS\_TYPE;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 LIST\_OF\_GDP : out GDP\_IDS\_LIST\_OF);

---

INQUIRE GENERALIZED DRAWING PRIMITIVE Уровень 0a  
 УЗНАТЬ ХАРАКТЕРИСТИКИ ОБОБЩЕННОГО  
 ПРИМИТИВА ВЫВОДА

procedure INQ\_GDP  
 (TYPE\_OF\_WS : in WS\_TYPE;  
 GDP : in GDP\_ID;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 LIST\_OF\_ATTRIBUTES\_USED : out ATTRIBUTES\_USED\_LIST\_OF);

INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES Уровень 0a  
 УЗНАТЬ ДЛИНУ ТАБЛИЦ, ХАРАКТЕРИЗУЮЩИХ СТАНЦИЮ

```

  procedure INQ_MAX_LENGTH_OF_WS_STATE_TABLES
    (TYPE_OF_WS           : in WS_TYPE;
     ERROR_INDICATOR     : out ERROR_NUMBER;
     MAX_POLYLINE_ENTRIES : out NATURAL;
     MAX_POLYMARKER_ENTRIES : out NATURAL;
     MAX_TEXT_ENTRIES    : out NATURAL;
     MAX_FILL_AREA_ENTRIES : out NATURAL;
     MAX_PATTERN_INDICES : out NATURAL;
     MAX_COLOUR_INDICES  : out NATURAL);
  
```

INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED Уровень 1a  
 УЗНАТЬ ДОПУСТИМОЕ ЧИСЛО ПРИОРИТЕТОВ СЕГМЕНТОВ

```

  procedure INQ_NUMBER_OF_SEGMENT_PRIORITIES_SUPPORTED
    (TYPE_OF_WS           : in WS_TYPE;
     ERROR_INDICATOR     : out ERROR_NUMBER;
     NUMBER_OF_PRIORITIES : out NATURAL);
  
```

INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES Уровень 1a  
 УЗНАТЬ СПОСОБ ДИНАМИЧЕСКОЙ МОДИФИКАЦИИ АТТРИБУТОВ СЕГМЕНТОВ

```

  procedure INQ_DYNAMIC_MODIFICATION_OF_SEGMENT_ATTRIBUTES
    (TYPE_OF_WS           : in WS_TYPE;
     ERROR_INDICATOR     : out ERROR_NUMBER;
     TRANSFORMATION_VISIBLE_TO_INVISIBLE : out DYNAMIC_MODIFICATION;
     TRANSFORMATION_INVISIBLE_TO_VISIBLE : out DYNAMIC_MODIFICATION);
  
```

HIGHLIGHTING	: out DYNAMIC_
	_MODIFICATION;
PRIORITY	: out DYNAMIC_
	_MODIFICATION;
ADDING_PRIMITIVES	: out DYNAMIC_
	_MODIFICATION;
DELETION_VISIBLE	: out DYNAMIC_
	_MODIFICATION);

INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES Уровень 0b

УЗНАТЬ ЧИСЛО ДОПУСТИМЫХ УСТРОЙСТВ ВВОДА

procedure INQ\_NUMBER\_OF\_AVAILABLE\_LOGICAL\_INPUT\_DEVICES

(TYPE_OF_WS	: in WS_TYPE;
ERROR_INDICATOR	: out ERROR_NUMBER;
LOCATOR	: out NATURAL;
STROKE	: out NATURAL;
VALUATOR	: out NATURAL;
CHOICE	: out NATURAL;
PICK	: out NATURAL;
STRING	: out NATURAL);

INQUIRE DEFAULT LOCATOR DEVICE Уровень 0b

DATA

УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ

УСТРОЙСТВА ВВОДА ПОЗИЦИИ

procedure INQ\_DEFAULT\_LOCATOR\_DEVICE\_DATA

(TYPE_OF_WS	: in WS_TYPE;
DEVICE	: in LOCATOR_DEVICE_
	NUMBER;
ERROR_INDICATOR	: out ERROR_NUMBER;
INITIAL_POSITION	: out WC.POINT;
LIST_OF_PROMPT_	: out LOCATOR_PROMPT_
_FCHO_TYPES	_ECHO_TYPES.LIST_OF;
ECHO_AREA	: out DC.RECTANGLE_LIMITS;
DATA_RECORD	: out LOCATOR_DATA_
	RECORD);

INQUIRE DEFAULT STROKE DEVICE Уровень 0b

DATA

УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ

УСТРОЙСТВА ВВОДА ПОСЛЕДОВАТЕЛЬНОСТИ

ПОЗИЦИИ

procedure INQ\_DEFAULT\_STROKE\_DEVICE\_DATA  
 (TYPE\_OF\_WS : in WS\_TYPE;  
 DEVICE : in STROKE\_DEVICE\_NUMBER;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 MAX\_BUFFER\_SIZE : out NATURAL;  
 LIST\_OF\_PROMPT\_ECHO\_TYPES : out STROKE\_PROMPT\_ECHO\_TYPES.LIST\_OF;  
 ECHO\_AREA : out DC\_RECTANGLE\_LIMITS;  
 DATA\_RECORD : out STROKE\_DATA\_RECORD);

INQUIRE DEFAULT VALUATOR DEVICE DATA Уровень 0b

УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ  
 УСТРОЙСТВА ВВОДА ЧИСЛА

procedure INQ\_DEFAULT\_VALUATOR\_DEVICE\_DATA  
 (TYPE\_OF\_WS : in WS\_TYPE;  
 DEVICE : in VALUATOR\_DEVICE\_NUMBER;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 INITIAL\_VALUE : out VALUATOR\_INPUT\_VALUE;  
 LIST\_OF\_PROMPT\_ECHO\_TYPES : out VALUATOR\_PROMPT\_ECHO\_TYPES.LIST\_OF;  
 ECHO\_AREA : out DC\_RECTANGLE\_LIMITS;  
 DATA\_RECORD : out VALUATOR\_DATA\_RECORD);

INQUIRE DEFAULT CHOICE DEVICE DATA Уровень 0b

УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ  
 УСТРОЙСТВА ВЫБОРА

procedure INQ\_DEFAULT\_CHOICE\_DEVICE\_DATA  
 (TYPE\_OF\_WS : in WS\_TYPE;  
 DEVICE : in CHOICE\_DEVICE\_NUMBER;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 MAX\_CHOICES : out CHOICE\_VALUE;  
 LIST\_OF\_PROMPT\_ECHO\_TYPES : out CHOICE\_PROMPT\_ECHO\_TYPES.LIST\_OF;  
 ECHO\_AREA : out DC\_RECTANGLE\_LIMITS;  
 DATA\_RECORD : out CHOICE\_DATA\_RECORD);



VISIBILITY	: out SEGMENT_VISIBILITY;
HIGHLIGHTING	: out SEGMENT_ _HIGHLIGHTING;
PRIORITY	: out SEGMENT_PRIORITY;
DETECTABILITY	: out SEGMENT_ _DETECTABILITY);

INQUIRE PIXEL ARRAY DIMENSIONS Уровень 0a  
 УЗНАТЬ РАЗМЕРНОСТЬ МАТРИЦЫ ПИКСЕЛЕЙ

procedure INQ\_PIXEL\_ARRAY\_DIMENSIONS  
 (WS : in WS\_ID;  
 CORNER\_I\_I : in WC.POINT;  
 CORNER\_DX\_DY : in WC.POINT;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 DIMENSIONS : out RASTER\_UNIT\_SIZE);

INQUIRE PIXEL ARRAY Уровень 0a  
 УЗНАТЬ МАТРИЦУ ПИКСЕЛЕЙ

procedure INQ\_PIXEL\_ARRAY  
 (WS : in WS\_ID;  
 CORNER : in WC.POINT;  
 DX : in RASTER\_UNITS;  
 DY : in RASTER\_UNITS;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 INVALID\_VALUES : out INVALID\_VALUES\_  
 \_INDICATOR;  
 PIXEL\_ARRAY : out VARIABLE\_PIXEL\_  
 \_COLOUR\_MATRIX);

INQUIRE\_PIXEL Уровень 0a  
 УЗНАТЬ ЦВЕТ ПИКСЕЛЯ

procedure INQ\_PIXEL  
 (WS : in WS\_ID;  
 POINT : in WC.POINT;  
 ERROR\_INDICATOR : out ERROR\_NUMBER;  
 PIXEL\_COLOUR : out PIXEL\_COLOUR\_INDEX);

INQUIRE INPUT QUEUE OVERFLOW Уровень 0c  
 УЗНАТЬ НАЛИЧИЕ ПЕРЕПОЛНЕНИЯ ОЧЕРЕДИ СОБЫТИЙ

procedure INQ\_INPUT\_QUEUE\_OVERFLOW  
 (ERROR\_INDICATOR : out ERROR\_NUMBER;  
 WS : in WS\_ID;

CLASS : out INPUT\_QUEUE\_CLASS;  
 DEVICE : out EVENT\_OVERFLOW\_  
           \_DEVICE\_NUMBER);

EVALUATE TRANSFORMATION MATRIX Уровень 1a

СФОРМИРОВАТЬ МАТРИЦУ ПРЕОБРАЗОВАНИЙ

```
procedure EVALUATE_TRANSFORMATION_MATRIX
(FIXED_POINT      : in WC.POINT;
SHIFT_VECTOR     : in WC.VECTOR;
ROTATION_ANGLE   : in RADIANS;
SCALE_FACTORS    : in TRANSFORMATION_
                 _FACTOR;
TRANSFORMATION   : out TRANSFORMATION_
                 _MATRIX);
```

```
procedure EVALUATE_TRANSFORMATION_MATRIX
(FIXED_POINT      : in NDC.POINT;
SHIFT_VECTOR     : in NDC.VECTOR;
ROTATION_ANGLE   : in RADIANS;
SCALE_FACTORS    : in TRANSFORMATION_
                 _FACTOR;
TRANSFORMATION   : out TRANSFORMATION_
                 _MATRIX);
```

ACCUMULATE TRANSFORMATION Уровень 1a

MATRIX

ВЫЧИСЛИТЬ РЕЗУЛЬТИРУЮЩУЮ МАТРИЦУ

ПРЕОБРАЗОВАНИЙ

```
procedure ACCUMULATE_TRANSFORMATION_MATRIX
(SOURCE_TRANSFOR- : in TRANSFORMATION_
MATION            : in TRANSFORMATION_
                 _MATRIX);
```

```
FIXED_POINT      : in WC.POINT;
SHIFT_VECTOR     : in WC.VECTOR;
ROTATION_ANGLE   : in RADIANS;
SCALE_FACTORS    : in TRANSFORMATION_
                 _FACTOR;
```

```
RESULT_TRANSFOR- : out TRANSFORMATION_
MATION            : in TRANSFORMATION_
                 _MATRIX);
```

```
procedure ACCUMULATE_TRANSFORMATION_MATRIX
(SOURCE_TRANSFOR- : in TRANSFORMATION_
MATION            : in TRANSFORMATION_
                 _MATRIX);
```

```
FIXED_POINT      : in NDC.POINT;
SHIFT_VECTOR     : in NDC.VECTOR;
ROTATION_ANGLE   : in RADIANS;
```

SCALE_FACTORS	: in TRANSFORMATION_	
		_FACTOR;
RESULT_TRANSFORMATION	: out TRANSFORMATION_	
		_MATRIX);
<hr/>		
EMERGENCY_CLOSE_GKS; АВАРИЙНО ЗАКРЫТЬ ЯГС procedure EMERGENCY_CLOSE_GKS;		Уровень 0a
<hr/>		
ERROR_HANDLING ОБРАБОТАТЬ ОШИБКУ procedure ERROR_HANDLING (ERROR_INDICATOR : in ERROR_NUMBER; GKS_FUNCTION : in STRING; ERROR_FILE : in STRING := DEFAULT_		Уровень 0a
		_ERROR_FILE);
<hr/>		
ERROR_LOGGING ЗАРЕГИСТРИРОВАТЬ ОШИБКУ procedure ERROR_LOGGING (ERROR_INDICATOR : out ERROR_NUMBER; GKS_FUNCTION : in STRING; ERROR_FILE : in STRING := DEFAULT_		Уровень 0a
		_ERROR_FILE);

## 5.2. Дополнительные функции

\* 5.2.1. Подпрограммы для манипуляции записями входных данных

В данном разделе определены функции и процедуры, которые необходимы для построения и запроса записей входных данных, декларированных как личные типы в данной связке, для всех шести классов устройств, определенных спецификацией ЯГС. Процедуры, представленные здесь, используют для построения записей данных для каждого зарегистрированного типа подсказки и эха. Также предоставляются соответствующие функции, позволяющие прикладным программам анализировать части записей данных, которые определены в ЯГС. Любую специфическую для реализации информацию в записях данных поддерживают личной и недоступной. Если любую из приведенных ниже процедур используют некорректно, то происходит исключительное событие GKS\_ERROR. Таким образом, если недопустимый тип подсказки и эха используют для построения процедур, то в файле ошибок регистрируют ошибку номер 2500.

Данные подпрограммы требуются на уровне 0b.

Для создания зависящих от реализации и зарегистрированных элементов реализация может предоставить дополнительные совмещаемые версии процедур BUILD и дополнительные функции для выделения информации из личных записей данных.

- Операции над записями данных устройства ввода позиций  

```
procedure BUILD_LOCATOR_DATA_RECORD
(PROMPT_ECHO_TYPE : in LOCATOR_PROMPT_
                                _ECHO_TYPE;
 DATA_RECORD      : out LOCATOR_DATA_
                                _RECORD);
```
- Создает и возвращает записи данных устройства ввода позиций.
- Операции над записями данных устройства ввода последовательности позиций  

```
procedure BUILD_STROKE_DATA_RECORD
(PROMPT_ECHO_TYPE : in STROKE_PROMPT_ECHO_
                                _TYPE;
 BUFFER_SIZE       : in POSITIVE;
 DATA_RECORD      : out STROKE_DATA_RECORD);
```
- Создает и возвращает записи данных устройства ввода последовательности позиций

```
function BUFFER_SIZE (DATA_RECORD : in STROKE_DATA_
                                _RECORD) return POSITIVE;
```

- Возвращает размер входного буфера устройства ввода последовательности позиций
- Операции над записями данных устройства ввода числа  

```
procedure BUILD_VALUATOR_DATA_RECORD
(PROMPT_ECHO_TYPE : in VALUATOR_PROMPT_
                                _ECHO_TYPE;
 LOW_VALUE         : in VALUATOR_INPUT_
                                _VALUE;
 HIGH_VALUE        : in VALUATOR_INPUT_
                                _VALUE;
 DATA_RECORD      : out VALUATOR_DATA_
                                _RECORD);
```
- Создает и возвращает записи данных устройства ввода числа.  

```
function HIGH_VALUE (DATA_RECORD : in VALUATOR_
                                _DATA_RECORD)
return VALUATOR_INPUT_VALUE;
```

— Возвращает наибольшее значение числа, запомненное в записи данных оценки.

```
function LOW-VALUE (DATA-RECORD : in VALUATOR-
                    _DATA-RECORD)
    return VALUATOR-INPUT-VALUE;
```

— Возвращает наименьшее значение числа, запомненное в записи данных оценки.

— Операции над записями данных устройства выбора

```
procedure BUILD-CHOICE-DATA-RECORD
    (PROMPT-ECHO-TYPE : in CHOICE-PROMPT-ECHO-
     _TYPE;
     DATA-RECORD : out CHOICE-DATA-RECORD);
```

— Создает и возвращает запись данных устройства выбора.

— Операции над записями данных устройства указания

```
procedure BUILD-PICK-DATA-RECORD
    (PROMPT-ECHO-TYPE : in PICK-PROMPT-ECHO-
     _TYPE;
     DATA-RECORD : out PICK-DATA-RECORD);
```

— Создает и возвращает запись данных устройства указания.

— Операции над записями данных устройства ввода строки

```
procedure BUILD-STRING-DATA-RECORD
    (PROMPT-ECHO-TYPE : in STRING-PROMPT-ECHO-
     _TYPE;
```

```
    INPUT-BUFFER-SIZE : in NATURAL;
```

```
    INITIAL-CURSOR-
    _POSITION : in NATURAL;
```

```
    DATA-RECORD : out STRING-DATA-RECORD);
```

— Создает и возвращает запись данных устройства ввода строки.

```
function INPUT-BUFFER-SIZE (DATA-RECORD:
                             in STRING-DATA-RECORD)
    return NATURAL;
```

— Возвращает размер буфера, используемого для запоминания вводимой строки, хранящейся в записи данных строки.

```
function INITIAL-CURSOR-POSITION (DATA-RECORD:
                                   in STRING-DATA-RECORD)
    return NATURAL;
```

— Возвращает начальную позицию курсора для вводимой строки, запомненной в записи данных строки.

## 5.2.2. Пакет обобщенной координатной системы ЯГС

Обобщенный пакет, декларируемый в данном разделе, представляет собой спецификацию обобщенной декартовой системы

координат для GKS. Данный пакет встречается три раза в пакете GKS\_TYPE для мировых координат, нормализованных координат устройства и координат устройства. Пакет определяет представление POINT (точки), POINT-ARRAY (матрицы точек), VECTOR (вектора) и RECTANGLE-LIMITS (прямоугольные ограничения) для координатной системы. Также определяется тип MAGNITUDE для измерения длин в координатной системе. Тип SIZE измеряет длины параллельно обоим осям, а тип RANGE-OF-MAGNITUDES определяет две длины внутри координатной системы: минимум и максимум для таких величин, как диапазон высот литер, доступных на устройстве. Данный обобщенный пакет включается в пакет типов GKS.

```

package
generic
  type COORDINATE_COMPONENT_TYPE is digits <>;
package GKS_COORDINATE_SYSTEM is
  type POINT is
    record
      X : COORDINATE_COMPONENT_TYPE;
      Y : COORDINATE_COMPONENT_TYPE;
    end record;
  type POINT_ARRAY is array (POSITIVE range <>)
    of POINT;
  type POINT_LIST (LENGTH : SMALL_NATURAL := 0) is
    record
      POINTS : POINT_ARRAY (1 .. LENGTH);
    end record;
  type VECTOR is new POINT;
  type RECTANGLE_LIMITS is
    record
      XMIN : COORDINATE_COMPONENT_TYPE;
      XMAX : COORDINATE_COMPONENT_TYPE;
      YMIN : COORDINATE_COMPONENT_TYPE;
      YMAX : COORDINATE_COMPONENT_TYPE;
    end record;
  type MAGNITUDE_BASE_TYPE is digits PRECISION;
  subtype MAGNITUDE is MAGNITUDE_BASE_TYPE range
    COORDINATE_COMPONENT_TYPE'SAFE_SMALL ..
    COORDINATE_COMPONENT_TYPE'SAFE_LARGE;
  type SIZE is
    record
      XAXIS : MAGNITUDE;
      YAXIS : MAGNITUDE;

```

```

end record;
type RANGE_OF_MAGNITUDES is
record
  MIN : MAGNITUDE;
  MAX : MAGNITUDE;
end record;
end GKS_COORDINATE_SYSTEM;

```

### 5.2.3. Общий пакет списка утилит ЯГС

Общий пакет `GSK_LIST_UTILITIES` встречается несколько раз в пакете `GSK_TYPE` для задания различных типов списков `LIST_OF` и подпрограмм для манипуляции ими. Каждый тип списка содержит различные значения типов элементов.

Тип списка декларируется как личный тип в `GSK_LIST_UTILITIES` для ограничения операций над типами списков, которые доступны внешним программным единицам. Декларация личных типов списков включает дискриминантную часть, которая определяет текущий размер списка. Объекты списков декларируются как неограниченные объекты, чтобы позволить динамическую модификацию размера списка.

Объект списка представляет собой последовательность значений типов элементов. Каждое значение типа элемента связывается с индексом. Значение индекса начинается с 1 и идет с приращением 1.

Размер объекта списка — это число значений типов элементов, запомненных в нем. Одно значение типа элемента может быть запомнено более одного раза внутри объекта списка. Объект списка может быть пустым. Максимальный размер объекта списка дан в общем параметре `MAX_LIST_SIZE`. Если данный параметр не задан, то используют принимаемое по умолчанию и зависящее от реализации значение.

— Подпрограммы обработки `LIST_OF`

```
function NULL_LIST return LIST_OF;
```

— Данная функция возвращает пустой объект `LIST_OF`. Данный список предназначен главным образом для тех, кто занимается реализацией `GKS`.

```
procedure ADD_TO_LIST
```

```

(ELEMENT      : in ELEMENT_TYPE;
 LIST        : in out LIST_OF);

```

— Данная процедура осуществляет запоминание значения параметра элемента в объекте параметра списка и увеличение размера списка на единицу. Индексное значение, равное увеличенному размеру списка, связывается с запомненным значением элемента.

Данная процедура порождает ошибку ЯГС 2502, если она вызывается, когда параметр списка имеет размер, равный максимальному. По желанию пользователю может гарантироваться незапоминание дублированных значений. Это реализуется вызовом ADD\_TO\_LIST с конкретным значением элемента, если функция IS\_IN\_LIST возвратила FALSE (ложь) для данного элемента.

```
procedure DELETE_FROM_LIST
(ELEMENT          in ELEMENT_TYPE;
LIST              : in out LIST_OF);
```

— Если объект параметра списка не содержит значения параметра элемента, данная процедура ничего не делает. В противном случае первый встретившийся элемент с данным значением удаляется. Размер объекта списка уменьшается на единицу, а индексы, связанные с оставшимися элементами, настраиваются таким образом, чтобы начинаться с единицы и идти с интервалом в единицу. При желании пользователь может удалить все элементы с данным значением. Это реализуется за счет повторения вызовов функции DELETE\_FROM\_LIST с конкретным значением элемента до тех пор, пока функция IS\_IN\_LIST возвращает TRUE (истина) для значения элемента.

```
function SIZE_OF_LIST (LIST : in LIST_OF)
return NATURAL;
```

— Данная функция возвращает число значений типов элементов, запомненных в объекте списка.

```
function IS_IN_LIST
(ELEMENT          : in ELEMENT_TYPE;
LIST              : in LIST_OF) return BOOLEAN;
```

— Данная функция возвращает TRUE (истина), если значение параметра элемента существует в объекте списка; в противном случае возвращается FALSE.

```
function LIST_ELEMENT
(INDEX            : in POSITIVE;
LIST              : in LIST_OF) return ELEMENT
TYPE;
```

— Данная функция возвращает значение элемента объекта списка, имеющего значение индекса, равное параметру индекса. Если параметр индекса превосходит текущий размер списка, генерируется ошибка ЯГС 2502.

```
function LIST (VALUES : in LIST_VALUES) return LIST_OF;
```

— Данная функция возвращает достоверный объект LIST\_OF. Если параметр VALUE является пустой матрицей, то возвращается пустой объект LIST\_OF. Если параметр не нулевой, данная

функция возвращает объект LIST\_OF, содержащий все значения в параметре VALUES. Если число значений элементов превосходит максимальный размер объекта LIST\_OF, генерируется ошибка ЯГС 2502.

— Спецификация обобщенного пакета

generic

```

type ELEMENT_TYPE is private;
MAX_LIST_SIZE : POSITIVE := implementation_defined;
package GKS_LIST_UTILITIES is
  subtype LIST_SIZE is NATURAL range 0..MAX_LIST_SIZE;
  type LIST_OF (SIZE : LIST_SIZE := 0) is private;
  type LIST_VALUES is array (POSITIVE range <>) of
    ELEMENT_TYPE;

```

function NULL\_LIST return LIST\_OF;

function SIZE\_OF\_LIST (LIST : in LIST\_OF) return NATURAL;

function IS\_IN\_LIST (ELEMENT

LIST

: in LIST\_OF) return BOOLEAN;

function LIST\_ELEMENT

(INDEX

LIST

: in POSITIVE;

: in LIST\_OF) return  
ELEMENT\_TYPE;

function LIST (VALUES

: in LIST VALUES) return

LIST\_OF;

procedure ADD\_TO\_LIST

(ELEMENT

LIST

: in ELEMENT\_TYPE;

: in out LIST\_OF);

procedure DELETE\_FROM

LIST (ELEMENT

LIST

: in ELEMENT\_TYPE;

: in out LIST\_OF);

private

— Декларация типа LIST\_OF зависит от реализации. Однако операции, неявно объявленные декларацией LIST\_OF, включая как присваивание, так и сравнение на равенство и неравенство, должны выполняться правильно. Данное требование предотвращает использование типов доступа для реализации типа LIST\_OF. Рекомендуемая реализация представлена ниже:

```

type LIST_OF (SIZE : LIST_SIZE := 0) is
  record

```

ELEMENTS : LIST\_VALUES (1..SIZE);

end\_record;

— Отметим, что декларирование неуточненных объектов LIST\_OF с использованием значения дискриминанта, принимаемого по умолчанию, допускает динамическую модификацию размера матрицы элементов.

```
end GKS_LIST_UTILITIES;
```

#### 5.2.4. Утилиты функций метафайла

Записи данных элементов метафайла являются сложными, для данных записей рекомендовано более 55 различных форматов. Прикладные программисты также могут определить новые форматы. Длина этих записей переменная. Записи данных могут содержать списки указателей, строки символов, матрицы индексов цветов и данные GDP и ESC. Длина записи зависит от числа элементов данных. GKS определяет, что формат зависит от реализации.

Тип записи данных должен быть личным, чтобы позволить непосредственную обработку содержимого записей.

Прикладной программист должен иметь возможность записать в метафайл неграфические данные. Это может быть предоставлено разрешением вывода символьных строк. Числовые данные могут быть преобразованы в символьные строки прикладным программистом до вызова функции BUILD\_NEW\_DATA\_RECORD построить новую запись данных метафайла GKSM.

### BUILD NEW GKSM DATA RECORD

```
procedure BUILD_NEW_GKSM_DATA_RECORD
```

```
  (TYPE_OF_ITEM           : in GKSM_ITEM_TYPE;
   ITEM_DATA              : in STRING;
   ITEM                   : out GKSM_DATA_RECORD);
```

```
ITEM DATA RECORD STRING
```

```
function ITEM_DATA_RECORD_STRING
```

```
  (ITEM : in GKSM_DATA_RECORD) return STRING;
```

#### 5.3. Настраиваемые варианты

Так как подмножества или расширения Ады не допускаются, то нет настраиваемых вариантов ЯГС/Ада. Более того, данное связывание не требует, чтобы необходимые для поддержки свойства были независимыми от реализации.

## СПЕЦИФИКАЦИЯ СКОМПИЛИРОВАННОГО ЯГС

(Данное приложение не является составной частью стандарта, а предоставляет дополнительную информацию)

```
with GKS_LIST_UTILITIES,
package GKS_TYPE is
```

— Данный пакет содержит все определения типов данных, используемые для задания связывания Ады с ЯГС. Данная компиляция была выполнена на компьютере MicroVax 2, используя компилятор VaxAda версии T1.4—32. Значения типов или подтипов, зависящих от реализации, были выбраны так, чтобы работать в среде 32-разрядного миникомпьютера с виртуальной памятью. Эти значения, возможно, потребуется изменить для микрокомпьютеров или машин с фиксированным размером памяти.

— Последующие константы являются зависящими от реализации и задают максимумы для реализации для типов ЯГС/Ада.

```
PRECISION                : constant      := 6;
SMALL_NATURAL_MAX       : constant      := 500;
STRING_SMALL_NATURAL_
_MAX                    : constant      := 100;
CHOICE_SMALL_NATURAL_
_MAX                    : constant      := 5;
```

```
subtype SMALL_NATURAL is NATURAL
range 0..SMALL_NATURAL_MAX;
```

— Это зависящие от реализации подтипы, допускающие применение для объектов записей различных типов записей без возникновения прерывания STORAGE\_ERROR.

```
subtype STRING_SMALL_NATURAL is NATURAL
range 0..STRING_SMALL_NATURAL_MAX;
```

— Это декларация зависящего от реализации подтипа, которая для объектов неуточненных записей допускает применение различных определенных ниже типов записей строк без возникновения прерывания STORAGE\_ERROR.

```
subtype CHOICE_SMALL_NATURAL is NATURAL
range 0..CHOICE_SMALL_NATURAL_MAX;
```

— Это декларация зависящего от реализации подтипа, который для объектов неуточненных записей допускает применение типа CHOICE\_PROMPT\_STRING\_LIST без возникновения прерывания STORAGE\_ERROR.

— Система координат ЯГС

```
generic
type COORDINATE_COMPONENT_TYPE is digits <>;
```

```
package GKS_COORDINATE_SYSTEM is
type POINT is
```

```
record
```

```
X: COORDINATE_COMPONENT_TYPE;
```

```
Y: COORDINATE_COMPONENT_TYPE;
```

```

        end record;
type POINT_ARRAY is array (POSITIVE range <>) of POINT;
type POINT_LIST (LENGTH : SMALL_NATURAL := 0) is
    record
        POINTS : POINT_ARRAY (1..LENGTH);
    end record;
type VECTOR is new POINT;
type RECTANGLE_LIMITS is
    record
        XMIN : COORDINATE_COMPONENT_TYPE;
        XMAX : COORDINATE_COMPONENT_TYPE;
        YMIN : COORDINATE_COMPONENT_TYPE;
        YMAX : COORDINATE_COMPONENT_TYPE;
    end record;
type MAGNITUDE_BASE_TYPE is digits PRECISION;
subtype MAGNITUDE is MAGNITUDE_BASE_TYPE range
    COORDINATE_COMPONENT_TYPE'SAFE_SMALL..
    COORDINATE_COMPONENT_TYPE'SAFE_LARGE;
type SIZE is
    record
        XAXIS : MAGNITUDE;
        YAXIS : MAGNITUDE;
    end record;
type RANGE_OF_MAGNITUDES is
    record
        MIN : MAGNITUDE;
        MAX : MAGNITUDE;
    end record;
end GKS_COORDINATE_SYSTEM;
— ASF Уровень 0а
type ASF is (BUNDLED, INDIVIDUAL);
— Данный тип определяет флаг выборки атрибутов, чье значение указывает,
откуда будут устанавливаться атрибуты примитива: из таблицы связей или из
индивидуального атрибута.
— ASF_LIST Уровень 0а
type ASF_LIST is
    record
        TYPE_OF_LINE_ASF : ASF;
        WIDTH_ASF : ASF;
        LINE_COLOUR_ASF : ASF;
        TYPE_OF_MARKER_ASF : ASF;
        SIZE_ASF : ASF;
        MARKER_COLOUR_ASF : ASF;
        FONT_PRECISION_ASF : ASF;
        EXPANSION_ASF : ASF;
        SPACING_ASF : ASF;
        TEXT_COLOUR_ASF : ASF;
        INTERIOR_ASF : ASF;
    end record;

```

```

STYLE_ASF : ASF;
FILL_AREA_COLOUR_ : ASF;
_ASF
end record;

```

— Запись, содержащая все флаги выборки атрибутов, с компонентами, обозначающими индивидуальный флаг.

```

— ATTRIBUTES_USED_TYPE
type ATTRIBUTES_USED_TYPE is
(POLYLINE_ATTRIBUTES,
POLYMARKER_ATTRIBUTES,
TEXT_ATTRIBUTES,
FILL_AREA_ATTRIBUTES);

```

Уровень 0a

— Типы атрибутов, которые могут быть использованы при генерации вывода для ОПВ и при генерации подсказки и эха различных типов для различных классов входных устройств

```

— ATTRIBUTES_USED
package ATTRIBUTES_USED is
new GKS_LIST_UTILITIES (ATTRIBUTES_USED_TYPE);

```

Уровень 0a

— Предназначен для списка используемых атрибутов

```

— SCALE_FACTOR
package SCALE_FACTOR_TYPE is

```

Уровень 0a

— Данный пакет используют для выделения производного типа SCALE\_FACTOR, так как его используют как базу для ряда других производных типов. В языке Ада, если база производного типа сама является производным типом, Этот тип предка не может быть декларирован непосредственно в видимой части того же самого пакета.

```

type SCALE_FACTOR is digits PRECISION;
— Тип, используемый для безразмерных масштабов.
end SCALE_FACTOR_TYPE;
use SCALE_FACTOR_TYPE;

```

```

— CHAR_EXPANSION
type CHAR_EXPANSION is new SCALE_FACTOR range
SCALE_FACTOR'SAFE_SMALL..SCALE_FACTOR'LAST;

```

Уровень 0a

— Определяет масштаб расширения литер. Масштаб должен быть безразмерным и больше нуля.

```

— CHAR_SPACING
type CHAR_SPACING is new SCALE_FACTOR;

```

Уровень 0a

— Определяет межлитерный просвет. Положительное значение указывает на величину дополнительного пространства между литерами в текстовой строке, а отрицательное — величину перекрытия прямоугольников литер в строке текста.

```

— DEVICE_NUMBER
package DEVICE_NUMBER_TYPE is
type DEVICE_NUMBER is new POSITIVE;

```

Уровень 0b

— Логическое устройство ввода устанавливают по номеру устройства.

```

end DEVICE_NUMBER_TYPE;
use DEVICE_NUMBER_TYPE;

```

```

— CHOICE_DEVICE_NUMBER
type CHOICE_DEVICE_NUMBER is new DEVICE_NUMBER;

```

Уровень 0b

— Определяет идентификатор устройства выбора.

- LOCATOR\_DEVICE\_NUMBER Уровень 0b  
type LOCATOR\_DEVICE\_NUMBER is new DEVICE\_NUMBER;  
— Предоставляет идентификатор устройства ввода позиции.
- PICK\_DEVICE\_NUMBER Уровень 1b  
type PICK\_DEVICE\_NUMBER is new DEVICE\_NUMBER;  
— Предоставляет идентификаторы устройства указания.
- STRING\_DEVICE\_NUMBER Уровень 0b  
type STRING\_DEVICE\_NUMBER is new DEVICE\_NUMBER;  
— Предоставляет номер устройства ввода строки.
- STROKE\_DEVICE\_NUMBER Уровень 0b  
type STROKE\_DEVICE\_NUMBER is new DEVICE\_NUMBER;  
— Предоставляет номер устройства ввода числа.
- VALUATOR\_DEVICE\_NUMBER Уровень 0b  
type VALUATOR\_DEVICE\_NUMBER is new DEVICE\_NUMBER;  
— Предоставляет идентификаторы устройства выбора.
- CHOICE\_PROMPT Уровень 0b  
type CHOICE\_PROMPT is (PFF, ON);  
— Определяет, будет или нет отображаться заданная подсказка для типа подсказки и эха устройства выбора.
- CHOICE\_PROMPTS Уровень 0b  
package CHOICE\_PROMPTS is  
  new GKS\_LIST\_UTILITIES (CHOICE\_PROMPT);  
— Предоставляет список подсказок.
- CHOICE\_PROMPT\_ECHO\_TYPE Уровень 0b  
type CHOICE\_PROMPT\_ECHO\_TYPE is new INTEGER;  
— Определяет тип подсказки и эха устройства выбора.
- CHOICE\_PROMPT\_ECHO\_TYPES Уровень 0b  
package CHOICE\_PROMPT\_ECHO\_TYPES is  
  new GKS\_LIST\_UTILITIES (CHOICE\_PROMPT\_ECHO\_TYPE);  
— Предоставляет список типов подсказок и эха устройства выбора.
- CHOICE\_PROMPT\_STRING Уровень 0b  
type CHOICE\_PROMPT\_STRING  
  (LENGTH : STRING\_SMALL\_NATURAL := 0) is  
  record  
    CONTENTS : STRING (1..LENGTH);  
  end record;  
— Предоставляет подсказки переменной длины. Объекты данного типа должны быть декларированы неуточненными, чтобы позволить динамическую модификацию длины.
- CHOICE\_PROMPT\_STRING\_ARRAY Уровень 0b  
type CHOICE\_PROMPT\_STRING\_ARRAY is array  
  (POSITIVE range <>) of CHOICE\_PROMPT\_STRING;  
— Предоставляет матрицу строк подсказок.
- CHOICE\_PROMPT\_STRING\_LIST Уровень 0b  
type CHOICE\_PROMPT\_STRING\_LIST (LENGTH :

CHOICE\_SMALL\_NATURAL := 0)

is record

LIST : CHOICE\_PROMPT\_STRING\_ARRAY (1..LENGTH);

end record;

— Предоставляет список строк подсказок.

— CHOICE\_REQUEST\_STATUS

Уровень 0b

type CHOICE\_REQUEST\_STATUS is (OK, NOCHOICE, NONE);

— Определяет статус для входных операций выбора для функций запроса.

— CHOICE\_STATUS

Уровень 0b

subtype CHOICE\_STATUS is CHOICE\_REQUEST\_STATUS range

OK..NOCHOICE;

— Указывает сделанный оператором выбор для функций опроса, получения события и справочных функций.

— CHOICE\_VALUE

Уровень 0b

type CHOICE\_VALUE is new POSITIVE;

— Определяет альтернативы, имеющиеся в реализации.

— CLIPPING\_INDICATOR

Уровень 0a

type CLIPPING\_INDICATOR is (CLIP, NOCLIP);

— Указывает на то, что будет или нет выполняться усечение.

— COLOUR\_AVAILABLE

Уровень 0a

type COLOUR\_AVAILABLE is (COLOUR, MONOCHROME);

— Указывает, имеется ли цветной вывод на станции.

— PIXEL\_COLOUR\_INDEX

Уровень 0a

type PIXEL\_COLOUR\_INDEX is new INTEGER

range -1..INTEGER'LAST;

— Тип цвета пиксела, где -1 обозначает неверный индекс цвета.

— COLOUR\_INDEX

Уровень 0a

subtype COLOUR\_INDEX is PIXEL\_COLOUR\_INDEX

range 0..PIXEL\_COLOUR\_INDEX'LAST;

— Предназначен для индексов в таблицах цветов.

— COLOUR\_INDICES

Уровень 0a

package COLOUR\_INDICES is new GKS\_LIST\_UTILITIES (COLOUR\_INDEX);

— Предоставляет набор индексов цветов, которые имеются на конкретной станции.

— COLOUR\_MATRIX

Уровень 0a

type COLOUR\_MATRIX is array (POSITIVE range &lt;&gt;, POSITIVE range &lt;&gt;) of COLOUR\_INDEX;

— Предоставляет матрицы, содержащие индексы цветов, соответствующие матрице ячеек или матрице шаблонов.

— INTENSITY

Уровень 0a

type INTENSITY is digits PRECISION range 0.0..1.0,

— Определяет область возможных интенсивностей цвета.

— COLOUR\_REPRESENTATION

Уровень 0a

type COLOUR\_REPRESENTATION is

record

RED : INTENSITY;

GREEN : INTENSITY;

BLUE : INTENSITY;

end record;

- Определяет представление цвета, как комбинацию интенсивностей в системе цветов красный—зеленый—голубой.
- CONTROL\_FLAG Уровень 0a  
 type CONTROL\_FLAG is (CONDITIONALLY, ALWAYS);  
 — Флаг управления используют, чтобы указать условия, при которых носитель изображения очищается
- DC\_TYPE Уровень 0a  
 type DC\_TYPE is digits PRECISION;  
 — Тип координат в системе координат устройства.
- DC Уровень 0a  
 package DC is new GKS\_COORDINATE\_SYSTEM (DC\_TYPE);  
 — Определяет систему координат устройства.
- DC\_UNITS Уровень 0a  
 type DC\_UNITS is (METRES, OTHER);  
 — Единицей измерения координат устройства для конкретной станции должен быть метр, если устройство не способно порождать масштабированные образы, или зависящая от конкретной станции единица в противном случае.
- DEFFERAL\_MODE Уровень 0a  
 type DEFFERAL\_MODE is (ASAP, BNIG, BNIL, ASTI);  
 — Определяет четыре отложенных режима ЯГС
- DISPLAY\_CLASS Уровень 0a  
 type DISPLAY\_CLASS is (VECTOR\_DISPLAY,  
 RASTER\_DISPLAY,  
 OTHER\_DISPLAY);  
 — Классификация станций категорий OUTPUT или OUTIN.
- DISPLAY\_SURFACE\_EMPTY Уровень 0a  
 type DISPLAY\_SURFACE\_EMPTY is (EMPTY, NOTEMPTY);  
 — Обозначает пуст ли носитель изображения.
- DYNAMIC\_MODIFICATION Уровень 1a  
 type DYNAMIC\_MODIFICATION is (IRG, IMM);  
 — Указывает, что обновление списка состояний выполняется немедленно или требует неявной повторной генерации.
- ECHO\_SWITCH Уровень 0b  
 type ECHO\_SWITCH is (ECHO, NOECHO);  
 — Обозначает, выполняется или нет выход эха.
- ERROR\_NUMBER Уровень 0a  
 type is ERROR\_NUMBER is new INTEGER;  
 — Определяет тип для значения индикатора ошибок
- INPUT\_CLASS Уровень 0b  
 type INPUT\_CLASS is (NONE,  
 LOCATOR\_INPUT,  
 STROKE\_INPUT,  
 VALUATOR\_INPUT,  
 CHOICE\_INPUT,  
 PICK\_INPUT,  
 STRING\_INPUT);  
 — Задаёт классификации входных устройств для станции категории INPUT или OUTIN.

- **EVENT\_DEVICE\_NUMBER** Уровень 0a  
 type **EVENT\_DEVICE\_NUMBER** (CLASS : INPUT.CLASS) : = NONE) is  
 record  
   case CLASS is  
     when NONE = > null;  
     when LOCATOR\_INPUT = > LOCATOR\_EVENT\_DEVICE  
       : LOCATOR\_DEVICE\_NUMBER;  
     when STROKE\_INPUT = > STROKE\_EVENT\_DEVICE  
       : STROKE\_DEVICE\_NUMBER;  
     when VALUATOR\_INPUT = > VALUATOR\_EVENT\_DEVICE  
       : VALUATOR\_DEVICE\_NUMBER;  
     when CHOICE\_INPUT = > CHOICE\_EVENT\_DEVICE  
       : CHOICE\_DEVICE\_NUMBER;  
     when PICK\_INPUT = > PICK\_EVENT\_DEVICE  
       : PICK\_DEVICE\_NUMBER;  
     when STRING\_INPUT = > STRING\_EVENT\_DEVICE  
       : STRING\_DEVICE\_NUMBER;  
   end case;  
 end record;  
 — Предназначен для возврата номера устройства любого класса из очереди событий.
- **INPUT\_QUEUE\_CLASS** Уровень 0a  
 subtype **INPUT\_QUEUE\_CLASS** is INPUT.CLASS range  
 LOCATOR\_INPUT..STRING\_INPUT;  
 — Определяет классификации устройств ввода для ситуаций, в которых классификация невозможна.
- **EVENT\_OVERFLOW\_DEVICE\_NUMBER** Уровень 0a  
 type **EVENT\_OVERFLOW\_DEVICE\_NUMBER**  
 (CLASS : INPUT\_QUEUE\_CLASS : = LOCATOR\_INPUT) is  
 record  
   case CLASS is  
     when LOCATOR\_INPUT = > LOCATOR\_EVENT\_DEVICE  
       : LOCATOR\_DEVICE\_NUMBER;  
     when STROKE\_INPUT = > STROKE\_EVENT\_DEVICE  
       : STROKE\_DEVICE\_NUMBER;  
     when VALUATOR\_INPUT = > VALUATOR\_EVENT\_DEVICE  
       : VALUATOR\_DEVICE\_NUMBER;  
     when CHOICE\_INPUT = > CHOICE\_EVENT\_DEVICE  
       : CHOICE\_DEVICE\_NUMBER;  
     when PICK\_INPUT = > PICK\_EVENT\_DEVICE  
       : PICK\_DEVICE\_NUMBER;  
     when STRING\_INPUT = > STRING\_EVENT\_DEVICE  
       : STRING\_DEVICE\_NUMBER;  
   end case;  
 end record;
- **FILL\_AREA\_INDEX** Уровень 0a  
 type **FILL\_AREA\_INDEX** is new POSITIVE;  
 — Определяет индексы таблицы связей областей заполнения.

- INTERIOR\_STYLE Уровень 0a  
 type INTERIOR\_STYLE is (HOLLOW, SOLID, PATTERN, HATCH);  
 — Определяет вид заполнения области.
- STYLE\_INDEX Уровень 0a  
 type STYLE\_INDEX is new INTEGER;  
 — Индекс вида — это либо HATCH\_STYLE, либо PATTERN\_STYLE.
- FILL\_AREA\_INDICES Уровень 0a  
 package FILL\_AREA\_INDICES is  
   new GKS\_LIST\_UTILITIES (FILL\_AREA\_INDEX);  
 — Предоставляет списки индексов таблицы связей областей заполнения.
- GDP\_ID Уровень 0a  
 type GDP\_ID is new INTEGER;  
 — Выбирает среди классов обобщенных примитивов вывода.
- GDP\_IDS Уровень 0a  
 package GDP\_IDS is new GKS\_LIST\_UTILITIES (GDP\_ID);  
 — Предоставляет списки идентификаторов обобщенных примитивов вывода.
- GKS\_LEVEL Уровень 0a  
 type GKS\_Level is (L0a, L0b, L0c, L1a, L1b, L1c, L2a, L2b, L2c);  
 — Доступные уровни ЯГС
- GKSM\_ITEM\_TYPE Уровень 0a  
 type GKSM\_ITEM\_TYPE is new NATURAL;  
 — Тип элемента, содержащегося в метафайле ЯГС
- HATCH\_STYLE Уровень 0a  
 subtype HATCH\_STYLE is STYLE\_INDEX;  
 — Определяет вид штриховки при заполнении области.
- HATCH\_STYLES Уровень 0a  
 package HATCH\_STYLES is new GKS\_LIST\_UTILITIES (HATCH\_STYLE);  
 — Предоставляет список видов штриховки.
- HORIZONTAL\_ALIGNMENT Уровень 0a  
 type HORIZONTAL\_ALIGNMENT is (NORMAL, LEFT, CENTRE, RIGHT);  
 — Выравнивание параллелограмма текста по отношению к горизонтальному положению текста.
- IMPLEMENTATION\_DEFINED\_ERROR Уровень 0a  
 subtype IMPLEMENTATION\_DEFINED\_ERROR is ERROR\_NUMBER  
   range ERROR\_NUMBER'FIRST..-1;  
 — Определяет область номеров ошибок, чтобы указывать, что произошла заданная реализацией ошибка.
- INPUT\_STATUS Уровень 0b  
 type INPUT\_STATUS is (OK, NONE).  
 — Определяет статус операции.
- INPUT\_STRING Уровень 0b  
 type INPUT\_STRING (LENGTH : STRING\_SMALL\_NATURAL := 0) is  
   record  
     CONTENTS : STRING (1..LENGTH);  
   end record;

— Предоставляет строку переменной длины. Объекты данного типа должны быть декларированы как неуточненные, чтобы позволить динамическую модификацию длины.

— INTERIOR\_STYLES Уровень 0a  
package INTERIOR\_STYLES is

new GKS\_LIST\_UTILITIES (INTERIOR\_STYLE);

— Предоставляет список видов заполнения.

— INVALID\_VALUES\_INDICATOR Уровень 0a  
type INVALID\_VALUES\_INDICATOR is (ABSENT, PRESENT);

— Указывает, имеются ли —1 в параметре PIXEL\_ARRAY, возвращаемом INQ\_PIXEL\_ARRAY.

— LANGUAGE\_BILDING\_ERROR Уровень 0a  
subtype LANGUAGE\_BILDING\_ERROR is ERROR\_NUMBER

range 2500..2999;

— Определяет область номеров ошибок, относящихся к ошибкам связывания с языком.

— POLYLINE\_INDEX Уровень 0a  
type POLYLINE\_INDEX is new POSITIVE;

— Определяет область индексов ломаной.

— LINETYPE Уровень 0a  
type LINETYPE is new INTEGER;

— Определяет типы линий, предоставляемых ЯГС.

— LINEWIDTH Уровень 0a  
type LINEWIDTH is new SCALE\_FACTOR range 0.0..

SCALE\_FACTOR'LAST;

— Ширина линии определяется масштабом толщины.

— LINETYPES Уровень 0a  
package LINETYPES is new GKS\_LIST\_UTILITIES (LINETYPE);

— Предоставляет список типов линий.

— LOCATOR\_PROMPT\_ECHO\_ Уровень 0b  
\_TYPE

type LOCATOR\_PROMPT\_ECHO\_TYPE is new INTEGER;

— Определяет типы подсказок и эха, поддерживаемых реализацией.

— LOCATOR\_PROMPT\_ECHO\_ Уровень 0b  
\_TYPES

package LOCATOR\_PROMPT\_ECHO\_TYPES is  
new GKS\_LIST\_UTILITIES (LOCATOR\_PROMPT\_ECHO\_TYPE);

— Предоставляет список типов подсказок и эха определителя местоположения.

— POLYMARKER\_INDEX Уровень 0a  
type POLYMARKER\_INDEX is new POSITIVE;

— Определяет область индексов таблицы связей полимаркеров.

— MARKER\_SIZE Уровень 0a  
type MARKER\_SIZE is new SCALE\_FACTOR range 0.0..

SCALE\_FACTOR'LAST;

— Размер маркера определяется масштабом.

— MARKER\_TYPE Уровень 0b  
type MARKER\_TYPE is new INTEGER;

— Определяет типы маркеров, предоставляемых ЯГС.

- MARKER\_DATA Уровень 0b
- MARKER\_TYPES Уровень 0a
- package MARKER\_TYPES is new GKS\_LIST\_UTILITIES  
(MARKER\_TYPE);
- Предоставляет список типов маркеров.
- MORE\_EVENTS Уровень 0a
- type MORE\_EVENTS is (NOMORE, MORE);
- Указывает, содержатся ли еще события в очереди событий.
- NDC\_TYPE Уровень та
- type NDC\_TYPE is digits PRECISION;
- Определяет тип координат в нормализованной системе координат.
- NDC Уровень та
- package NDC is new GKS\_COORDINATE\_SYSTEM (NDC\_TYPE);
- Задает нормализованную систему координат.
- NEW\_FRAME\_NECESSARY Уровень 0a
- type NEW\_FRAME\_NECESSARY is (NO, YES);
- Указывает, необходимы ли действия по новому коду при модификации.
- OPERATING\_MODE Уровень 0b
- type OPERATING\_MODE is (REQUEST\_MODE, SAMPLE\_MODE,  
EVENT\_MODE);
- Определяет режимы работы устройства ввода.
- OPERATING\_STATE Уровень 0a
- type OPERATING\_STATE is (GKCL, GKOP, WSOP, WSAC, SGOP);
- Определяет пять состояний ЯГС.
- PATTERN\_INDEX Уровень 0a
- subtype PATTERN\_INDEX is STYLE\_INDEX range 1.. STYLE\_INDEX'LAST;
- Определяет диапазон индексов таблицы шаблонов.
- PATTERN\_INDICES Уровень 0a
- package PATTERN\_INDICES is  
new GKS\_LIST\_UTILITIES (PATTERN\_INDEX);
- Предоставляет списки индексов таблицы шаблонов.
- PICK\_ID Уровень 1b
- type PICK\_ID is new POSITIVE;
- Определяет диапазон идентификаторов устройства указания, существующих в реализации.
- PICK\_IDS Уровень 1b
- package PICK\_IDS is new GKS\_LIST\_UTILITIES (PICK\_ID);
- Предоставляет списки идентификаторов устройства указания.
- PICK\_PROMPT\_ECHO\_TYPE Уровень 1b
- type PICK\_PROMPT\_ECHO\_TYPE is new INTEGER;
- Определяет тип подсказки и эха для устройства указания.
- PICK\_PROMPT\_ECHO\_TYPES Уровень 1b
- package PICK\_PROMPT\_ECHO\_TYPES is new GKS\_LIST\_UTILITIES  
(PICK\_PROMPT\_ECHO\_TYPE);
- Предоставляет списки типов подсказки и эха устройства указания.

- PICK\_REQUEST\_STATUS Уровень 1b  
 type PICK\_REQUEST\_STATUS is (OK, NOPICK, NONE);  
 — Определяет статус операции ввода указания для функции запроса.
- PICK\_STATUS Уровень 1b  
 subtype PICK\_STATUS is PICK\_REQUEST\_STATUS range OK..NOPICK;  
 — Определяет статус операции ввода указания для функций получения информации.
- PIXEL\_COLOUR\_MATRIX Уровень 0a  
 type PIXEL\_COLOUR\_MATRIX is array (POSITIVE range <>, POSITIVE range <>) of PIXEL\_COLOUR\_INDEX;  
 — Предоставляет матрицы цветов пикселей.
- POLYLINE\_INDICES Уровень 0a  
 package POLYLINE\_INDICES is new GKS\_LIST\_UTILITIES (POLYLINE\_INDEX);  
 — Предоставляет списки индексов ломаной.
- POLYMARKER\_INDICES Уровень 0a  
 package POLYMARKER\_INDICES is new GKS\_LIST\_UTILITIES (POLYMARKER\_INDEX);  
 — Предоставляет списки индексов полимаркеров.
- RADIANS Уровень 1a  
 type RADIANS is digits PRECISION;  
 — Величины, используемые в выполнении преобразования сегмента (угла вращения). Положительное значение указывает на вращение против часовой стрелки.
- RANGE\_OF\_EXPANSIONS Уровень 0a  
 type RANGE\_OF\_EXPANSIONS is  
 record  
 MIN : CHAR\_EXPANSION;  
 MAX : CHAR\_EXPANSION;  
 end record;  
 — Предоставляет область значений масштаба расширения литер.
- RASTER\_UNITS Уровень 0a  
 type RASTER\_UNITS is new POSITIVE;  
 — Определяет область единиц раstra.
- RASTER\_UNIT\_SIZE Уровень 0a  
 type RASTER\_UNIT\_SIZE is  
 record  
 X : RASTER\_UNITS;  
 Y : RASTER\_UNITS;  
 end record;  
 — Определяет размер экрана дисплея в растровых единицах.
- REGENERATION\_MODE Уровень 0a  
 type REGENERATION\_MODE is (SUPPRESSED, ALLOWED);  
 — Указывает, подавлена или разрешена неявная повторная генерация.
- RELATIVE\_PRIORITY Уровень 0a  
 type RELATIVE\_PRIORITY is (HIGHER, LOWER);  
 — Обозначает относительный приоритет между двумя преобразованиями нормирования.

- RETURN\_VALUE\_TYPE Уровень 0a  
type RETURN\_VALUE\_TYPE is (SET, REALIZED);  
— Указывает, возвращаемое значение следует рассматривать как заданное программой или как действительно реализованное устройством.
- SEGMENT\_DETECTABILITY Уровень 1a  
type SEGMENT\_DETECTABILITY is (UNDETECTABLE, DETECTABLE);  
— Указывает, является или нет сегмент обнаруживаемым.
- SEGMENT\_HIGHLIGHTING Уровень 1a  
type SEGMENT\_HIGHLIGHTING is (NORMAL, HIGHLIGHTED);  
— Указывает, является или нет сегмент выделяемым.
- SEGMENT\_NAME Уровень 1a  
type SEGMENT\_NAME is new POSITIVE;  
— Определяет диапазон имен сегментов.
- SEGMENT\_NAMES Уровень 1a  
package SEGMENT\_NAMES is new GKS\_LIST\_UTILITIES  
(SEGMENT\_NAME);  
— Дает список имен сегментов.
- SEGMENT\_PRIORITY Уровень 1a  
type SEGMENT\_PRIORITY is digits PRECISION range 0.0..0.1;  
— Определяет приоритет сегмента.
- SEGMENT\_VISIBILITY Уровень 1a  
type SEGMENT\_VISIBILITY is (VISIBLE, INVISIBLE);  
— Обозначает, является ли сегмент видимым или нет.
- STRING\_PROMPT\_ECHO\_TYPE Уровень 0b  
type STRING\_PROMPT\_ECHO\_TYPE is new INTEGER;  
— Определяет типы подсказки и эха устройства ввода строки.
- STRING\_PROMPT\_ECHO\_TYPES Уровень 0b  
package STRING\_PROMPT\_ECHO\_TYPES is  
new GKS\_LIST\_UTILITIES (STRING\_PROMPT\_ECHO\_TYPE);  
— Предоставляет списки типов подсказок и эха устройства ввода строки.
- STROKE\_PROMPT\_ECHO\_TYPE Уровень 0b  
type STROKE\_PROMPT\_ECHO\_TYPE is new INTEGER;  
— Определяет типы подсказок и эха устройства ввода последовательности позиций.
- STROKE\_PROMPT\_ECHO\_TYPES Уровень 0b  
package STROKE\_PROMPT\_ECHO\_TYPES is  
new GKS\_LIST\_UTILITIES (STROKE\_PROMPT\_ECHO\_TYPE);  
— Предоставляет списки типов подсказок и эха ввода последовательности позиций.
- VERTICAL\_ALIGNMENT Уровень 0a  
type VERTICAL\_ALIGNMENT is (NORMAL, TOP, CAP, HALF, BASE,  
BOTTOM);  
— Выравнивание параллелограмма текста по отношению к вертикальной позиции текста.
- TEXT\_ALIGNMENT Уровень 0a  
type TEXT\_ALIGNMENT is  
record

**HORIZONTAL** : **HORIZONTAL\_ALIGNMENT**;  
**VERTICAL** : **VERTICAL\_ALIGNMENT**;  
 end record;

— Тип атрибута, управляющего позиционированием параллелограмма текста по отношению к позиции текста, имеющего горизонтальные и вертикальные компоненты, как определено выше.

— **WC\_TYPE** Уровень 0a  
 type **WC\_TYPE** is digits **PRECISION**;  
 — Определяет точность для типа мировых координат.

— **WC** Уровень 0a  
 package **WC** is new **GKS\_COORDINATE\_SYSTEM** (**WC\_TYPE**);  
 — Определяет мировую систему координат.

— **TEXT\_EXTENT** Уровень 0a  
**\_PARALLELOGRAM**  
 type **TEXT\_EXTENT\_PARALLELOGRAM** is  
 record

**LOWER\_LEFT** : **WC.POINT**;  
**LOWER\_RIGHT** : **WC.POINT**;  
**UPPER\_RIGHT** : **WC.POINT**;  
**UPPER\_LEFT** : **WC.POINT**;  
 end record;

— Определяет угловые точки параллелограмма текста по отношению к вертикальному позиционированию текста.

— **TEXT\_FONT** Уровень 0a  
 type **TEXT\_FONT** is new **INTEGER**;  
 — Определяет типы шрифтов, предоставляемых реализацией.

— **TEXT\_PRECISION** Уровень 0a  
 type **TEXT\_PRECISION** is (**STRING\_PRECISION**,  
**CHAR\_PRECISION**,  
**STROKE\_PRECISION**);  
 — Точность, с которой появляется текст.

— **TEXT\_FONT\_PRECISION** Уровень 0a  
 type **TEXT\_FONT\_PRECISION** is  
 record

**FONT** : **TEXT\_FONT**;  
**PRECISION** : **TEXT\_PRECISION**;  
 end record;

— Данный тип определяет запись, описывающую атрибут шрифта и точности текста.

— **TEXT\_FONT\_PRECISIONS** Уровень 0a  
 package **TEXT\_FONT\_PRECISIONS** is  
 new **GKS\_LIST\_UTILITIES** (**TEXT\_FONT\_PRECISION**);

— Предоставляет список пар шрифта и точности текста.

— **TEXT\_INDEX** Уровень 0a  
 type **TEXT\_INDEX** is new **POSITIVE**;

— Определяет диапазон значений индексов таблицы связей текста

— **TEXT\_INDICES** Уровень 0a  
 package **TEXT\_INDICES** is new **GKS\_LIST\_UTILITIES** (**TEXT\_INDEX**);  
 — Предоставляет списки индексов текста.

- **TEXT\_PATH** Уровень 0a  
 type TEXT\_PATH is (RIGHT, LEFT, UP, DOWN);  
 — Направление строки текста.
- **TRANSFORMATION\_FACTOR** Уровень 1a  
 type TRANSFORMATION\_FACTOR is  
   record  
     X : NDC\_TYPE;  
     Y : NDC\_TYPE;  
 end record;
- Масштаб, используемый в матрицах преобразования для выполнения преобразования сегментов.
- **TRANSFORMATION\_MATRIX** Уровень 1a  
 type TRANSFORMATION\_MATRIX is array (1..2, 1..3) of NDC\_TYPE;
- Для преобразований сегментов, отображающихся внутрь пространства НК.
- **TRANSFORMATION\_NUMBER** Уровень 0a  
 type TRANSFORMATION\_NUMBER is new NATURAL;  
 — Номер преобразования нормирования.  
 subtype POSITIVE\_TRANSFORMATION\_NUMBER is  
   TRANSFORMATION\_NUMBER  
   range 1.. TRANSFORMATION\_NUMBER'LAST;
- Номер преобразования нормирования, соответствующий установленному преобразованию.
- **TRANSFORMATION\_PRIORITY\_ARRAY** Уровень 0a  
 type TRANSFORMATION\_PRIORITY\_ARRAY is array  
   (POSITIVE range <>) of TRANSFORMATION\_NUMBER;
- Тип для запоминания номера преобразования.
- **TRANSFORMATION\_PRIORITY\_LIST** Уровень 0a  
 type TRANSFORMATION\_PRIORITY\_LIST (LENGTH : SMALL\_NATURAL  
   = 0) is  
   record  
     CONTENTS : TRANSFORMATION\_PRIORITY\_ARRAY (1.. LENGTH);  
 end record;
- Предоставляет упорядоченный по приоритетам список номеров преобразований.
- **UPDATE\_REGENERATION\_FLAG** Уровень 0a  
 type UPDATE\_REGENERATION\_FLAG is (PERFORM, POSTRONE);  
 — Флаг повторной генерации на дисплее.
- **UPDATE\_STATE** Уровень 0a  
 type UPDATE\_STATE is (NOTPENDING, PENDING);  
 — Указывает, было ли запрошено изменение преобразования станции.
- **VALUATOR\_INPUT\_VALUE** Уровень 0b  
 type VALUATOR\_INPUT\_VALUE is digits PRECISION;  
 — Определяет точность вводимых величин в данной реализации.
- **VALUATOR\_PROMPT\_ECHO\_TYPE** Уровень 0b  
 type VALUATOR\_PROMPT\_ECHO\_TYPE is new INTEGER;  
 — Определяет возможные типы подсказок и эха устройства ввода числа.

- VALUATOR\_PROMPT\_ECHO\_TYPES Уровень 0b  
 package VALUATOR\_PROMPT\_ECHO\_TYPES is  
 new GKS\_LIST\_UTILITIES (VALUATOR\_PROMPT\_ECHO\_TYPE);
- Предоставляет списки типов подсказок и эха устройства ввода числа.
- VARIABLE\_COLOUR\_MATRIX Уровень 0a  
 type VARIABLE\_COLOUR\_MATRIX (DX : SMALL\_NATURAL := 0;  
 DY : SMALL\_NATURAL := 0) is  
 record  
 MATRIX : COLOUR\_MATRIX (1..DX, 1..DY);  
 end record;
- Предоставляет матрицы переменного размера, содержащие индексы цветов, соответствующие матрице ячеек или матрице шаблонов.
- VARIABLE\_CONNECTION\_ID Уровень 0a  
 type VARIABLE\_CONNECTION\_ID  
 (LENGTH : STRING\_SMALL\_NATURAL := 0) is  
 record  
 CONNECT : STRING (1..LENGTH);  
 end record;
- Определяет идентификатор связи переменной длины для INQ\_WS\_CONNECTION\_AND\_TYPE.
- VARIABLE\_PIXEL\_COLOUR\_MATRIX Уровень 0a  
 type VARIABLE\_PIXEL\_COLOUR\_MATRIX (DX : SMALL\_NATURAL := 0;  
 DY : SMALL\_NATURAL := 0) is  
 record  
 MATRIX : PIXEL\_COLOUR\_MATRIX (1..DX, 1..DY);  
 end record;
- Предоставляет матрицы переменного размера для цветов пикселей.
- WS\_CATEGORY Уровень 0a  
 type WS\_CATEGORY is (OUTPUT, INPUT, OUTIN, WISS, MO, MI);  
 — Тип для категорий станций ЯГС.
- WS\_ID Уровень 0a  
 type WS\_ID in new POSITIVE;  
 — Определяет область идентификаторов станций.
- WS\_IDS Уровень 0a  
 package WS\_IDS is new GKS\_LIST\_UTILITIES (WS\_ID);  
 — Предоставляет списки идентификаторов станций.
- WS\_STATES Уровень 0a  
 type WS\_STATES is (INACTIVE, ACTIVE);  
 — Состояние станции.
- WS\_TYPES Уровень 0a  
 type WS\_TYPES is new POSITIVE;  
 — Область значений, соответствующих достоверным типам станций. Константы, определяющие имена для различных типов станций, должны предоставляться реализацией.
- WS\_TYPES Уровень 0a  
 package WS\_TYPES is new GKS\_LIST\_UTILITIES (WS\_TYPE);  
 — Предоставляет списки типов станций.

— INDIVIDUAL\_ATTRIBUTE\_VALUES

Уровень 0a

type INDIVIDUAL\_ATTRIBUTE\_VALUES is

record

TYPE_OF_LINE	: LINETYPE;
WIDTH	: LINEWIDTH;
LINE_COLOUR	: COLOUR_INDEX;
TYPE_OF_MARKER	: MARKER_TYPE;
SIZE	: MARKER_SIZE;
MARKER_COLOUR	: COLOUR_INDEX;
PONT_PRECISION	: TEXT_FONT_PRECISION;
EXPANSION	: CHAR_EXPANSION;
SPACING	: CHAR_SPACING;
TEXT_COLOUR	: COLOUR_INDEX;
INTERIOR	: INTERIOR_STYLE;
STYLE	: STYLE_INDEX;
FILL_AREA_COLOUR	: COLOUR_INDEX;
ASF	: ASF_LIST;

end record;

— Запись, содержащая текущие индивидуальные атрибуты для процедуры INQ\_CURRENT\_INDIVIDUAL\_ATTRIBUTE\_VALUES.

— PRIMITIVE\_ATTRIBUTE\_VALUES

Уровень 0a

type PRIMITIVE\_ATTRIBUTE\_VALUES is

record

INDEX_POLYLINE	: POLYLINE_INDEX;
INDEX_POLYMARKER	: POLYMARKER_INDEX;
INDEX_TEXT	: TEXT_INDEX;
CHAR_HEIGHT	: WC.MAGNITUDE;
CHAR_UP_VECTOR	: WC.VECTOR;
CHAR_WIDTH	: WC.MAGNITUDE;
CHAR_BASE_VECTOR	: WC.VECTOR;
PATH	: TEXT_PATH;
ALIGNMENT	: TEXT_ALIGNMENT;
INDEX_FILL_AREA	: FILL_AREA_INDEX;
PATTERN_WIDTH_VECTOR	: WC.VECTOR;
PATTERN_HEIGHT_VECTOR	: WC.VECTOR;
PATTERN_REFERENCE	: WC.POINT;

POINT

end record;

— Запись, содержащая атрибуты текущего примитива для процедуры INQ\_CURRENT\_PRIMITIVE\_ATTRIBUTE\_VALUES.

— Далее задается предопределенное прерывание GKS-ERROR, заданное в п. 3.2.3.

GKS\_ERROR : exception;

— Далее идут декларации зависящих от реализации констант для определения типов ЯГС/Ада. Некоторые константы используют для задания значений параметров, принимаемых по умолчанию, процедурам ЯГС.

— Следующие константы определяют стандартные типы линий ЯГС:

SOLID_LINE	: constant LINETYPE := 1;
DASHED_LINE	: constant LINETYPE := 2;
DOTTED_LINE	: constant LINETYPE := 3;
DASHED_DOTTED_LINE	: constant LINETYPE := 4;

Следующие константы определяют стандартные типы маркеров ЯГС:

```

DOT_MARKER           : constant MARKER_TYPE := 1;
PLUS_MARKER          : constant MARKER_TYPE := 2;
STAR_MARKER          : constant MARKER_TYPE := 3;
ZERO_MARKER          : constant MARKER_TYPE := 4;
X_MARKER             : constant MARKER_TYPE := 5;

```

Следующие константы определяют типы подсказок и эха, поддерживаемые ЯГС:

```

DEFAULT_LOCATOR      : constant LOCATOR_PROMPT_ECHO_TYPE := 1;
CROSS_HAIR_LOCATOR  : constant LOCATOR_PROMPT_ECHO_TYPE := 2;
TRACKING_CROSS_LOCATOR : constant LOCATOR_PROMPT_ECHO_TYPE := 3;
RUBBER_BAND_LINE_LOCATOR : constant LOCATOR_PROMPT_ECHO_TYPE := 4;
RECTANGLE_LOCATOR   : constant LOCATOR_PROMPT_ECHO_TYPE := 5;
DIGITAL_LOCATOR     : constant LOCATOR_PROMPT_ECHO_TYPE := 6;

DEFAULT_STROKE       : constant STROKE_PROMPT_ECHO_TYPE := 1;
DIGITAL_STROKE       : constant STROKE_PROMPT_ECHO_TYPE := 2;
MARKER_STROKE        : constant STROKE_PROMPT_ECHO_TYPE := 3;
LINE_STROKE          : constant STROKE_PROMPT_ECHO_TYPE := 4;

DEFAULT_VALUATOR     : constant VALUATOR_PROMPT_ECHO_TYPE := 1;
GRAPHICAL_VALUATOR   : constant VALUATOR_PROMPT_ECHO_TYPE := 2;
DIGITAL_VALUATOR     : constant VALUATOR_PROMPT_ECHO_TYPE := 3;

DEFAULT_CHOICE_PROMPT_ECHO : constant CHOICE_PROMPT_ECHO_TYPE := 1;
CHOICE_PROMPT_ECHO   : constant CHOICE_PROMPT_ECHO_TYPE := 2;
STRING_PROMPT_CHOICE : constant CHOICE_PROMPT_ECHO_TYPE := 3;
STRING_INPUT_CHOICE  : constant CHOICE_PROMPT_ECHO_TYPE := 4;
SEGMENT_CHOICE       : constant CHOICE_PROMPT_ECHO_TYPE := 5;
DEFAULT_STRING       : constant STRING_PROMPT_ECHO_TYPE := 1;
DEFAULT_PICK         : constant PICK_PROMPT_ECHO_TYPE := 1;
GROUP_HIGHLIGHT_PICK : constant PICK_PROMPT_ECHO_TYPE := 2;
SEGMENT_HIGHLIGHT_PICK : constant PICK_PROMPT_ECHO_TYPE := 3;

```

— Следующие константы используют для задания принимаемых по умолчанию значений параметров процедур ЯГС.

```

DEFAULT_MEMORY_UNITS : constant := 0;

DEFAULT_ERROR_FILE : constant STRING := « »;
end GKS_TYPES;
— ПАКЕТ ЯГС

```

with GKS\_TYPES,  
use GKS\_TYPES,  
package GKS is

- Пакет ЯГС содержит все процедуры, которые требуются для реализации уровня 2с ЯГС.
- Последующие типы данных являются личными и включены в пакет ЯГС для легкости манипулирования.
- CHOICE\_DATA\_RECORD Уровень 0b  
 type CHOICE\_DATA\_RECORD (PROMPT\_ECHO\_TYPE :  
   'CHOICE\_PROMPT\_ECHO\_TYPE' = DEFAULT\_CHOICE) is private;  
 — Определяет запись для инициализации устройства выбора
- GKSM\_DATA\_RECORD Уровень 0b  
 type GKSM\_DATA\_RECORD (TYPE\_OF\_ITEM : GKSM\_ITEM\_TYPE := 0;  
   LENGTH : NATURAL := 0) is private;  
 — Запись данных для метафайла GKSM.
- LOCATOR\_DATA\_RECORD Уровень 0b  
 type LOCATOR\_DATA\_RECORD (PROMPT\_ECHO\_TYPE :  
   LOCATOR\_PROMPT\_ECHO\_TYPE := DEFAULT\_LOCATOR)  
   is private;  
 — Определяет запись для инициализации устройства ввода позиции.
- PICK\_DATA\_RECORD Уровень 1b  
 type PICK\_DATA\_RECORD (PROMPT\_ECHO\_TYPE :  
   PICK\_PROMPT\_ECHO\_TYPE := DEFAULT\_PICK) is private;  
 — Определяет запись для инициализации устройства указания.
- STRING\_DATA\_RECORD Уровень 0b  
 type STRING\_DATA\_RECORD (PROMPT\_ECHO\_TYPE :  
   STRING\_PROMPT\_ECHO\_TYPE := DEFAULT\_STRING) is private;  
 — Определяет запись для инициализации устройства ввода строки.
- STROKE\_DATA\_RECORD Уровень 0b  
 type STROKE\_DATA\_RECORD (PROMPT\_ECHO\_TYPE :  
   STROKE\_PROMPT\_ECHO\_TYPE := DEFAULT\_STROKE) is private;  
 — Определяет запись для инициализации устройства ввода последовательности позиций.
- VALUATOR\_DATA\_RECORD Уровень 0b  
 type VALUATOR\_DATA\_RECORD (PROMPT\_ECHO\_TYPE :  
   VALUATOR\_PROMPT\_ECHO\_TYPE := DEFAULT\_VALUATOR)  
   is private;  
 — Определяет запись для инициализации устройства ввода числа.  
 — Подпрограммы для манипулирования записями данных ввода.  
 — Процедуры и функции, определенные ниже, необходимы для построения и опроса записей входных данных, декларированных как личные типы в пакете для каждого из шести классов устройств ввода, определенных спецификацией ЯГС. Процедуры, представленные ниже, используют для построения записей данных для каждого из типов подсказок и эха устройств, применяемых для инициализации конкретных устройств ввода. Также предоставлены соответствующие функции, позволяющие прикладному программному обеспечению ЯГС/Ада проанализировать части записи данных, которые определены ЯГС. Любую специфическую для реализации информацию в записях данных поддерживают личной и недоступной. Прерывания GKS\_ERROR возникают, если любую из приведенных ниже процедур используют неправильно. Таким образом, если

применяют недопустимый тип подсказки и эха, в файле ошибок регистрируют ошибку номер 2500.

— Операции над записями данных устройства ввода позиций.

```
procedure BUILD_LOCATOR_DATA_RECORD
(PROMPT_ECHO_TYPE      : in LOCATOR_PROMPT_ECHO_TYPE;
 DATA_RECORD          : out LOCATOR_DATA_RECORD);
```

— Строит и возвращает запись данных устройства ввода позиций.

```
procedure BUILD_STROKE_DATA_RECORD
(PROMPT_ECHO_TYPE      : in STROKE_PROMPT_ECHO_TYPE;
 BUFFER_SIZE           : in POSITIVE;
 DATA_RECORD          : out STROKE_DATA_RECORD);
```

— Строит и возвращает запись данных устройства ввода последовательности позиций.

```
function BUFFER_SIZE(DATA_RECORD : in STROKE_DATA_RECORD)
return POSITIVE;
```

— Операции над записями данных устройства ввода числа.

```
procedure BUILD_VALUATOR_DATA_RECORD
(PROMPT_ECHO_TYPE      : in VALUATOR_PROMPT_ECHO_TYPE;
 LOW_VALUE              : in VALUATOR_INPUT_VALUE;
 HIGH_VALUE             : in VALUATOR_INPUT_VALUE;
 DATA_RECORD          : out VALUATOR_DATA_RECORD);
```

— Строит и возвращает запись данных устройства ввода числа.

```
function HIGH_VALUE (DATA_RECORD : in VALUATOR_DATA_RECORD)
return VALUATOR_INPUT_VALUE;
```

— Возвращает наибольшее число, запомненное в записи данных устройства ввода числа.

```
function LOW_VALUE (DATA_RECORD : in VALUATOR_DATA_RECORD)
return VALUATOR_INPUT_VALUE;
```

— Возвращает наименьшее число, запомненное в записи данных устройства ввода числа.

— Операции над записями данных устройства выбора.

```
procedure BUILD_CHOICE_DATA_RECORD
(PROMPT_ECHO_TYPE      : in CHOICE_PROMPT_ECHO_TYPE;
 DATA_RECORD          : out CHOICE_DATA_RECORD);
```

— Строит и возвращает запись данных устройства выбора.

— Операции над записями данных устройства указания.

```
procedure BUILD_PICK_DATA_RECORD
(PROMPT_ECHO_TYPE      : in PICK_PROMPT_ECHO_TYPE;
 DATA_RECORD          : out PICK_DATA_RECORD);
```

— Строит и возвращает запись данных устройства указания.

— Операции над записями данных устройства ввода строки.

```
procedure BUILD_STRING_DATA_RECORD
(PROMPT_ECHO_TYPE      : in STRING_PROMPT_ECHO_TYPE;
 INPUT_BUFFER_SIZE     : in POSITIVE;
```

INITIAL\_CURSOR\_POSITION : in NATURAL;  
 DATA\_RECORD : out STRING\_DATA\_RECORD);

— Строит и возвращает запись данных строк.

function INPUT\_BUFFER\_SIZE  
 (DATA\_RECORD : in STRING\_DATA\_RECORD) return NATURAL;

— Возвращает размер буфера, использованного для запоминания строки, размещенной в записи данных строки.

function INITIAL\_CURSOR\_POSITION  
 (DATA\_RECORD : in STRING\_DATA\_RECORD) return NATURAL;

— Возвращает начальную позицию курсора для устройства ввода строки, запомненной в записи данных устройства ввода строки.

— Процедуры ЯГС

— ФУНКЦИИ УПРАВЛЕНИЯ

procedure OPEN\_GKS  
 (ERROR\_FILE : in STRING := DEFAULT\_ERROR\_FILE;  
 AMOUNT\_OF\_MEMORY : in NATURAL := DEFAULT\_MEMORY\_UNITS);

procedure CLOSE\_GKS;

procedure OPEN\_WS  
 (WS : in WS\_ID;  
 CONNECTION : in STRING;  
 TYPE\_OF\_WS : in WS\_TYPE);

procedure CLOSE\_WS  
 (WS : in WS\_ID);

procedure ACTIVATE\_WS  
 (WS : in WS\_ID);

procedure DEACTIVATE\_WS  
 (WS : in WS\_ID);

procedure CLEAR\_WS  
 (WS : in WS\_ID);  
 FLAG : in CONTROL\_FLAG);

procedure REDRAW\_ALL\_SEGMENTS\_ON\_WS  
 (WS : in WS\_ID);

procedure UPDATE\_WS  
 (WS : in WS\_ID;  
 REGENERATION : in UPDATE\_REGENERATION\_FLAG);

procedure SET\_DEFERRAL\_STATE  
 (WS : in WS\_ID;  
 DEFERRAL : in DEFERRAL\_MODE;  
 REGENERATION : in REGENERATION\_MODE);

procedure MESSAGE  
 (WS : in WS\_ID;  
 CONTENTS : in STRING);

## — ФУНКЦИИ ВВОДА ГРАФИЧЕСКИХ ДАННЫХ

```

procedure POLYLINE
  (POINTS                : in WC.POINT_ARRAY);
procedure POLYMARKER
  (POINTS                : in WC.POINT_ARRAY);
procedure TEXT
  (POSITION              : in WC.POINT;
   CHAR_STRING           : in STRING);
procedure FILL_AREA
  (POINTS                : in WC.POINT_ARRAY);
procedure CELL_ARRAY
  (CORNER_1_1            : in WC.POINT;
   CORNER_DX_DY          : in WC.POINT;
   CELLS                 : in COLOUR_MATRIX);

```

## — ФУНКЦИИ ЗАДАНИЯ АТТРИБУТОВ ВЫХОДНЫХ ДАННЫХ

```

procedure SET_POLYLINE_INDEX
  (INDEX                 : in POLYLINE_INDEX);
procedure SET_LINETYPE
  (TYPE_OF_LINE         : in LINETYPE);
procedure SET_LINEWIDTH_SCALE_FACTOR
  (WIDTH                 : in LINEWIDTH);
procedure SET_POLYLINE_COLOUR_INDEX
  (LINE_COLOUR           : in COLOUR_INDEX);
procedure SET_POLYMARKER_INDEX
  (INDEX                 : in POLYMARKER_INDEX);
procedure SET_MARKER_TYPE
  (TYPE_OF_MARKER       : in MARKER_TYPE);
procedure SET_MARKER_SIZE_SCALE_FACTOR
  (SIZE                  : in MARKER_SIZE);
procedure SET_POLYMARKER_COLOUR_INDEX
  (MARKER_COLOUR        : in COLOUR_INDEX);
procedure SET_TEXT_INDEX
  (INDEX                 : in TEXT_INDEX);
procedure SET_TEXT_FONT_AND_PRECISION
  (FONT_PRECISION       : in TEXT_FONT_PRECISION);
procedure SET_CHAR_EXPANSION_FACTOR
  (EXPANSION             : in CHAR_EXPANSION);
procedure SET_CHAR_SPACING
  (SPACING               : in CHAR_SPACING);
procedure SET_TEXT_COLOUR_INDEX
  (TEXT_COLOUR           : in COLOUR_INDEX);
procedure SET_CHAR_HEIGHT
  (HEIGHT                : in WC.MAGNITUDE);

```

```

procedure SET_CHAR_UP_VECTOR
  (CHAR_UP_VECTOR      : in WC_VECTOR);
procedure SET_TEXT_PATH
  (PATH                : in TEXT_PATH);
procedure SET_TEXT_ALIGNMENT
  (ALIGNMENT          : in TEXT_ALIGNMENT);
procedure SET_FILL_AREA_INDEX
  (INDEX              : in FILL_AREA_INDEX);
procedure SET_FILL_AREA_INTERIOR_STYLE
  (INTERIOR           : in INTERIOR_STYLE);
procedure SET_FILL_AREA_STYLE_INDEX
  (STYLE              : in STYLE_INDEX);
procedure SET_FILL_AREA_COLOUR_INDEX
  (FILL_AREA_COLOUR  : in COLOUR_INDEX);
procedure SET_PATTERN_SIZE
  (SIZE               : in WC.SIZE);
procedure SET_PATTERN_REFERENCE_POINT
  (POINT              : in WC.POINT);
procedure SET_ASF
  (ASF                : in ASF.LIST);
procedure SET_PICK_ID
  (PICK               : in PICK_ID);
procedure SET_POLYLINE_REPRESENTATION
  (WS                  : in WS_ID;
   INDEX               : in POLYLINE_INDEX;
   TYPE_OF_LINE        : in LINETYPE;
   WIDTH               : in LINEWIDTH;
   LINE_COLOUR         : in COLOUR_INDEX);
procedure SET_POLYMARKER_REPRESENTATION
  (WS                  : in WS_ID;
   INDEX               : in POLYMARKER_INDEX;
   TYPE_OF_MARKER      : in MARKER_TYPE;
   SIZE                : in MARKER_SIZE;
   MARKER_COLOUR       : in COLOUR_INDEX);
procedure SET_TEXT_REPRESENTATION
  (WS                  : in WS_ID;
   INDEX               : in TEXT_INDEX;
   FONT_PRECISION      : in TEXT_FONT_PRECISION;
   EXPANSION           : in CHAR_EXPANSION;
   SPACING              : in CHAR_SPACING;
   TEXT_COLOUR         : in COLOUR_INDEX);
procedure SET_FILL_AREA_REPRESENTATION
  (WS                  : in WS_ID;
   INDEX               : in FILL_AREA_INDEX;
   INTERIOR             : in INTERIOR_STYLE;
   STYLE                : in STYLE_INDEX;
   FILL_AREA_COLOUR     : in COLOUR_INDEX);

```

procedure SET\_PATTERN\_REPRESENTATION

(WS : in WS\_ID;  
INDEX : in PATTERN\_INDEX;  
PATTERN : in COLOUR\_MATRIX);

procedure SET\_COLOUR\_REPRESENTATION

(WS : in WS\_ID;  
INDEX : in COLOUR\_INDEX;  
RGB\_COLOUR : in COLOUR\_REPRESENTATION);

— ФУНКЦИИ ПРЕОБРАЗОВАНИЯ

procedure SET\_WINDOW

(TRANSFORMATION : in POSITIVE\_TRANSFORMATION\_NUMBER;  
WINDOW\_LIMITS : in WC.RECTANGLE\_LIMITS);

procedure SET\_VIEWPORT

(TRANSFORMATION : in POSITIVE\_TRANSFORMATION\_NUMBER;  
VIEWPORT\_LIMITS : in NDC.RECTANGLE\_LIMITS);

procedure SET\_VIEWPORT\_INPUT\_PRIORITY

(TRANSFORMATION : in TRANSFORMATION\_NUMBER;  
REFERENCE : in TRANSFORMATION\_NUMBER;  
TRANSFORMATION\_PRIORITY : in RELATIVE\_PRIORITY);

procedure SELECT\_NORMALIZATION\_TRANSFORMATION

(TRANSFORMATION : in TRANSFORMATION\_NUMBER);

procedure SET\_CLIPPING\_INDICATOR

(CLIPPING : in CLIPPING\_INDICATOR);

procedure SET\_WS\_WINDOW

(WS : in WS\_ID;  
WS\_WINDOW\_LIMITS : in NDC.RECTANGLE\_LIMITS);

procedure SET\_WS\_VIEWPORT

(WS : in WS\_ID;  
WS\_VIEWPORT\_LIMITS : in DC.RECTANGLE\_LIMITS);

— ФУНКЦИИ СЕГМЕНТАЦИИ

procedure CREATE\_SEGMENT

(SEGMENT : in SEGMENT\_NAME);

procedure CLOSE\_SEGMENT;

procedure RENAME\_SEGMENT

(OLD\_NAME : in SEGMENT\_NAME;  
NEW\_NAME : in SEGMENT\_NAME);

procedure DELETE\_SEGMENT

(SEGMENT : in SEGMENT\_NAME);

procedure DELETE\_SEGMENT\_FROM\_WS

(WS : in WS\_ID;  
SEGMENT : in SEGMENT\_NAME);

procedure ASSOCIATE\_SEGMENT\_WITH\_WS

(WS : in WS\_ID;  
SEGMENT : in SEGMENT\_NAME);

```

procedure COPY_SEGMENT_TO_WS
  (WS
   SEGMENT
   : in WS_ID;
   : in SEGMENT_NAME);

procedure INSERT_SEGMENT
  (SEGMENT
   TRANSFORMATION
   : in SEGMENT_NAME;
   : in TRANSFORMATION_MATRIX);

procedure SET_SEGMENT_TRANSFORMATION
  (SEGMENT
   TRANSFORMATION
   : in SEGMENT_NAME;
   : in TRANSFORMATION_MATRIX);

procedure SET_VISIBILITY
  (SEGMENT
   VISIBILITY
   : in SEGMENT_NAME;
   : in SEGMENT_VISIBILITY);

procedure SET_HIGHLIGHTING
  (SEGMENT
   HIGHLIGHTING
   : in SEGMENT_NAME;
   : in SEGMENT_HIGHLIGHTING);

procedure SET_SEGMENT_PRIORITY
  (SEGMENT
   PRIORITY
   : in SEGMENT_NAME;
   : in SEGMENT_PRIORITY);

procedure SET_DETECTABILITY
  (SEGMENT
   DETECTABILITY
   : in SEGMENT_NAME;
   : in SEGMENT_DETECTABILITY);

— ФУНКЦИИ ВВОДА

procedure INITIALISE_LOCATOR
  (WS
   DEVICE
   INITIAL_TRANSFORMATION
   INITIAL_POSITION
   ECHO_AREA
   DATA_RECORD
   : in WS_ID;
   : in LOCATOR_DEVICE_NUMBER;
   : in TRANSFORMATION_NUMBER;
   : in WC_POINT;
   : in DC_RECTANGLE_LIMITS;
   : in LOCATOR_DATA_RECORD);

procedure INITIALISE_STROKE
  (WS
   DEVICE
   INITIAL_TRANSFORMATION
   INITIAL_STROKE
   ECHO_AREA
   DATA_RECORD
   : in WS_ID;
   : in STROKE_DEVICE_NUMBER;
   : in TRANSFORMATION_NUMBER;
   : in WC_POINT_ARRAY;
   : in DC_RECTANGLE_LIMITS;
   : in STROKE_DATA_RECORD);

procedure INITIALISE_VALUATOR
  (WS
   DEVICE
   INITIAL_VALUE
   ECHO_AREA
   DATA_RECORD
   : in WS_ID;
   : in VALUATOR_DEVICE_NUMBER;
   : in VALUATOR_INPUT_VALUE;
   : in DC_RECTANGLE_LIMITS;
   : in VALUATOR_DATA_RECORD);

procedure INITIALISE_CHOICE
  (WS
   DEVICE
   INITIAL_STATUS
   INITIAL_CHOICE
   ECHO_AREA
   DATA_RECORD
   : in WS_ID;
   : in CHOICE_DEVICE_NUMBER;
   : in CHOICE_STATUS;
   : in CHOICE_VALUE;
   : in DC_RECTANGLE_LIMITS;
   : in CHOICE_DATA_RECORD);

```

```

procedure INITIALISE_PICK
  (WS
  DEVICE
  INITIAL_STATUS
  INITIAL_SEGMENT
  INITIAL_PICK
  ECHO_AREA
  DATA_RECORD
  : in WS_ID;
  : in PICK_DEVICE_NUMBER;
  : in PICK_STATUS;
  : in SEGMENT_NAME;
  : in PICK_ID;
  : in DC.RECTANGLE_LIMITS;
  : in PICK_DATA_RECORD);

procedure INITIALISE_STRING
  (WS
  DEVICE
  INITIAL_STRING
  ECHO_AREA
  DATA_RECORD
  : in WS_ID;
  : in STRING_DEVICE_NUMBER;
  : in INPUT_STRING;
  : in DC.RECTANGLE_LIMITS;
  : in STRING_DATA_RECORD);

procedure SET_LOCATOR_MODE
  (WS
  DEVICE
  MODE
  SWITCH
  : in WS_ID;
  : in LOCATOR_DEVICE_NUMBER;
  : in OPERATING_MODE;
  : in ECHO_SWITCH);

procedure SET_STROKE_MODE
  (WS
  DEVICE
  MODE
  SWITCH
  : in WS_ID;
  : in STROKE_DEVICE_NUMBER;
  : in OPERATING_MODE;
  : in ECHO_SWITCH);

procedure SET_VALUATOR_MODE
  (WS
  DEVICE
  MODE
  SWITCH
  : in WS_ID;
  : in VALUATOR_DEVICE_NUMBER;
  : in OPERATING_MODE;
  : in ECHO_SWITCH);

procedure SET_CHOICE_MODE
  (WS
  DEVICE
  MODE
  SWITCH
  : in WS_ID;
  : in CHOICE_DEVICE_NUMBER;
  : in OPERATING_MODE;
  : in ECHO_SWITCH);

procedure SET_PICK_MODE
  (WS
  DEVICE
  MODE
  SWITCH
  : in WS_ID;
  : in PICK_DEVICE_NUMBER;
  : in OPERATING_MODE;
  : in ECHO_SWITCH);

procedure SET_STRING_MODE
  (WS
  DEVICE
  MODE
  SWITCH
  : in WS_ID;
  : in STRING_DEVICE_NUMBER;
  : in OPERATING_MODE;
  : in ECHO_SWITCH);

procedure REQUEST_LOCATOR
  (WS
  DEVICE
  STATUS
  TRANSFORMATION
  POSITION
  : in WS_ID;
  : in LOCATOR_DEVICE_NUMBER;
  : out INPUT_STATUS;
  : out TRANSFORMATION_NUMBER;
  : out WC.POINT);

```

```

procedure REQUEST_STROKE
  (WS
   DEVICE
   STATUS
   TRANSFORMATION
   STROKE_POINTS
  )
  : in WS_ID;
  : in STROKE_DEVICE_NUMBER;
  : out INPUT_STATUS;
  : out TRANSFORMATION_NUMBER;
  : out WC.POINT_LIST);

procedure REQUEST_VALUATOR
  (WS
   DEVICE
   STATUS
   VALUE
  )
  : in WS_ID;
  : in VALUATOR_DEVICE_NUMBER;
  : out INPUT_STATUS;
  : out VALUATOR_INPUT_VALUE);

procedure REQUEST_CHOICE
  (WS
   DEVICE
   STATUS
   CHOICE_NUMBER
  )
  : in WS_ID;
  : in CHOICE_DEVICE_NUMBER;
  : out CHOICE_REQUEST_STATUS;
  : out CHOICE_VALUE);

procedure REQUEST_PICK
  (WS
   DEVICE
   STATUS
   SEGMENT
   PICK
  )
  : in WS_ID;
  : in PICK_DEVICE_NUMBER;
  : out PICK_REQUEST_STATUS;
  : out SEGMENT_NAME;
  : out PICK_ID);

procedure REQUEST_STRING
  (WS
   DEVICE
   STATUS
   CHAR_STRING
  )
  : in WS_ID;
  : in STRING_DEVICE_NUMBER;
  : out INPUT_STATUS;
  : out INPUT_STRING);

procedure SAMPLE_LOCATOR
  (WS
   DEVICE
   TRANSFORMATION
   POSITION
  )
  : in WS_ID;
  : in LOCATOR_DEVICE_NUMBER;
  : out TRANSFORMATION_NUMBER;
  : out WC.POINT);

procedure SAMPLE_STROKE
  (WS
   DEVICE
   TRANSFORMATION
   STROKE_POINTS
  )
  : in WS_ID;
  : in STROKE_DEVICE_NUMBER;
  : out TRANSFORMATION_NUMBER;
  : out WC.POINT_LIST);

procedure SAMPLE_VALUATOR
  (WS
   DEVICE
   VALUE
  )
  : in WS_ID;
  : in VALUATOR_DEVICE_NUMBER;
  : out VALUATOR_INPUT_VALUE);

procedure SAMPLE_CHOICE
  (WS
   DEVICE
   STATUS
   CHOICE_NUMBER
  )
  : in WS_ID;
  : in CHOICE_DEVICE_NUMBER;
  : out CHOICE_STATUS;
  : out CHOICE_VALUE);

procedure SAMPLE_PICK
  (WS
   DEVICE
  )
  : in WS_ID;
  : in PICK_DEVICE_NUMBER;

```



```

procedure INTERPRET_ITEM
  (ITEM : in GKSM_DATA_RECORD);
— СПРАВОЧНЫЕ ФУНКЦИИ
procedure INQ_OPERATING_STATE_VALUE
  (VALUE : out OPERATING_STATE);
procedure INQ_LEVEL_OF_GKS
  (ERROR_INDICATOR : out ERROR_NUMBER;
  LEVEL : out GKS_LEVEL);
procedure INQ_LIST_OF_AVAILABLE_WS_TYPES
  (ERROR_INDICATOR : out ERROR_NUMBER;
  TYPES : out WS_TYPES.LIST_OF);
procedure INQ_WS_MAX_NUMBERS
  (ERROR_INDICATOR : out ERROR_NUMBER;
  MAX_OPEN_WS : out POSITIVE;
  MAX_ACTIVE_WS : out POSITIVE;
  MAX_SEGMENT_WS : out POSITIVE);
procedure INQ_MAX_NORMALIZATION_TRANSFORMATION_NUMBER
  (ERROR_INDICATOR : out ERROR_NUMBER;
  TRANSFORMATION : out TRANSFORMATION_NUMBER);
procedure INQ_SET_OF_OPEN_WS
  (ERROR_INDICATOR : out ERROR_NUMBER;
  WS : out WS_IDS.LIST_OF);
procedure INQ_SET_OF_ACTIVE_WS
  (ERROR_INDICATOR : out ERROR_NUMBER;
  WS : out WS_IDS.LIST_OF);
procedure INQ_CURRENT_PRIMITIVE_ATTRIBUTES_VALUES
  (ERROR_INDICATOR : out ERROR_NUMBER;
  ATTRIBUTES : out PRIMITIVE_ATTRIBUTES_VALUES);
procedure INQ_POLYLINE_INDEX
  (ERROR_INDICATOR : out ERROR_NUMBER;
  INDEX : out POLYLINE_INDEX);
procedure INQ_POLYMARKER_INDEX
  (ERROR_INDICATOR : out ERROR_NUMBER;
  INDEX : out POLYMARKER_INDEX);
procedure INQ_TEXT_INDEX
  (ERROR_INDICATOR : out ERROR_NUMBER;
  INDEX : out TEXT_INDEX);
procedure INQ_CHAR_HEIGHT
  (ERROR_INDICATOR : out ERROR_NUMBER;
  HEIGHT : out WC.MAGNITUDE);
procedure INQ_CHAR_UP_VECTOR
  (ERROR_INDICATOR : out ERROR_NUMBER;
  VECTOR : out WC.VECTOR);
procedure INQ_CHAR_WIDTH
  (ERROR_INDICATOR : out ERROR_NUMBER;
  WIDTH : out WC.MAGNITUDE);

```

```

procedure INQ_CHAR_BASE_VECTOR
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   VECTOR               : out WC.VECTOR);

procedure INQ_TEXT_PATH
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   PATH                 : out TEXT_PATH);

procedure INQ_TEXT_ALIGNMENT
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   ALIGNMENT           : out TEXT_ALIGNMENT);

procedure INQ_FILL_AREA_INDEX
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   INDEX               : out FILL_AREA_INDEX);

procedure INQ_PATTERN_WIDTH_VECTOR
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   WIDTH              : out WC.VECTOR);

procedure INQ_PATTERN_HEIGHT_VECTOR
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   VECTOR             : out WC.VECTOR);

procedure INQ_PATTERN_REFERENCE_POINT
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   REFERENCE_POINT     : out WC.POINT);

procedure INQ_CURRENT_PICK_ID_VALUE
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   PICK                : out PICK_ID);

procedure INQ_CURRENT_INDIVIDUAL_ATTRIBUTE_VALUES
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   ATTRIBUTES          : out INDIVIDUAL_ATTRIBUTE_
                           VALUES);

procedure INQ_LINETYPE
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   TYPE_OF_LINE        : out LINETYPE);

procedure INQ_LINEWIDTH_SCALE_FACTOR
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   WIDTH               : out LINEWIDTH);

procedure INQ_POLYLINE_COLOUR_INDEX
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   LINE_COLOUR         : out COLOUR_INDEX);

procedure INQ_POLYMARKER_TYPE
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   TYPE_OF_MARKER      : out MARKER_TYPE);

procedure INQ_POLYMARKER_SIZE_SCALE_FACTOR
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   SIZE               : out MARKER_SIZE);

procedure INQ_POLYMARKER_COLOUR_INDEX
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   MARKER_COLOUR       : out COLOUR_INDEX);

```

```

procedure INQ_TEXT_FONT_AND_PRECISION
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   FONT_PRECISION       : out TEXT_FONT_PRECISION);

procedure INQ_CHAR_EXPANSION_FACTOR
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   EXPANSION            : out CHAR_EXPANSION);

procedure INQ_CHAR_SPACING
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   SPACING              : out CHAR_SPACING);

procedure INQ_TEXT_COLOUR_INDEX
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   TEXT_COLOUR          : out COLOUR_INDEX);

procedure INQ_FILL_AREA_INTERIOR_STYLE
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   INTERIOR             : out INTERIOR_STYLE);

procedure INQ_FILL_AREA_STYLE_INDEX
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   STYLE                : out STYLE_INDEX);

procedure INQ_FILL_AREA_COLOUR_INDEX
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   FILL_AREA_COLOUR     : out COLOUR_INDEX);

procedure INQ_LIST_OF_ASF
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   LIST                 : out ASF_LIST);

procedure INQ_CURRENT_NORMALIZATION_TRANSFORMATION_NUMBER
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   TRANSFORMATION        : out TRANSFORMATION_NUMBER);

procedure INQ_LIST_OF_NORMALIZATION_TRANSFORMATION_NUMBERS
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   LIST                 : out TRANSFORMATION_PRIORITY_
                        _LIST);

procedure INQ_NORMALIZATION_TRANSFORMATION
  (TRANSFORMATION        : in TRANSFORMATION_NUMBER;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   WINDOW_LIMITS        : out WC.RECTANGLE_LIMITS;
   VIEWPORT_LIMITS      : out NDC.RECTANGLE_LIMITS);

procedure INQ_CLIPPING
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   CLIPPING              : out CLIPPING_INDICATOR;
   CLIPPING_RECTANGLE_  : out NDC.RECTANGLE_LIMITS);
LIMITS

procedure INQ_NAME_OF_OPEN_SEGMENT
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   SEGMENT               : out SEGMENT_NAME);

procedure INQ_SET_OF_SEGMENT_NAMES_IN_USE
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   SEGMENTS              : out SEGMENT_NAMES_LIST_OF);

```

```

procedure INQ_MORE_SIMULTANEOUS_EVENTS
  (ERROR_INDICATOR : out ERROR_NUMBER;
   EVENTS          : out MORE_EVENTS);

procedure INQ_WS_CONNECTION_AND_TYPE
  (WS              : in WS_ID;
   ERROR_INDICATOR : out ERROR_NUMBER;
   CONNECTION      : out VARIABLE_CONNECTION_ID;
   TYPE_OF_WS      : out WS_TYPE);

procedure INQ_WS_STATE
  (WS              : in WS_ID;
   ERROR_INDICATOR : out ERROR_NUMBER;
   STATE           : out WS_STATE);

procedure INQ_WS_DEFERRAL_AND_UPDATE_STATES
  (WS              : in WS_ID;
   ERROR_INDICATOR : out ERROR_NUMBER;
   DEFERRAL        : out DEFERRAL_MODE;
   REGENERATION    : out REGENERATION_MODE;
   DISPLAY         : out DISPLAY_SURFACE_EMPTY;
   FRAME_ACTION    : out NEW_FRAME_NECESSARY);

procedure INQ_LIST_OF_POLYLINE_INDICES
  (WS              : in WS_ID;
   ERROR_INDICATOR : out ERROR_NUMBER;
   INDICES         : out POLYLINE_INDICES_LIST_OF);

procedure INQ_POLYLINE_REPRESENTATION
  (WS              : in WS_ID;
   INDEX           : in POLYLINE_INDEX;
   RETURNED_VALUES : in RETURN_VALUE_TYPE;
   ERROR_INDICATOR : out ERROR_NUMBER;
   TYPE_OF_LINE    : out LINETYPE;
   WIDTH           : out LINEWIDTH;
   LINE_COLOUR     : out COLOUR_INDEX);

procedure INQ_LIST_OF_POLYMARKER_INDICES
  (WS              : in WS_ID;
   ERROR_INDICATOR : out ERROR_NUMBER;
   INDICES         : out POLYMARKER_INDICES_LIST_OF);

procedure INQ_POLYMARKER_REPRESENTATION
  (WS              : in WS_ID;
   INDEX           : in POLYMARKER_INDEX;
   RETURNED_VALUES : in RETURN_VALUE_TYPE;
   ERROR_INDICATOR : out ERROR_NUMBER;
   TYPE_OF_MARKER  : out MARKER_TYPE;
   SIZE            : out MARKER_SIZE;
   MARKER_COLOUR   : out COLOUR_INDEX);

procedure INQ_LIST_OF_TEXT_INDICES
  (WS              : in WS_ID;
   ERROR_INDICATOR : out ERROR_NUMBER;
   INDICES         : out TEXT_INDICES_LIST_OF);

procedure INQ_TEXT_REPRESENTATION
  (WS              : in WS_ID;

```

```

INDEX : in TEXT_INDEX;
RETURNED_VALUES : in RETURN_VALUE_TYPE;
ERROR_INDICATOR : out ERROR_NUMBER;
FONT_PRECISION : out TEXT_FONT_PRECISION;
EXPANSION : out CHAR_EXPANSION;
SPACING : out CHAR_SPACING;
TEXT_COLOUR : out COLOUR_INDEX);

procedure INQ_TEXT_EXTENT
  (WS : in WS_ID;
   POSITION : in WC_POINT;
   CHAR_STRING : in STRING;
   ERROR_INDICATOR : out ERROR_NUMBER;
   CONCATENATION_POINT : out WC_POINT;
   TEXT_EXTENT : out TEXT_EXTENT_PARALLELOGRAM);

procedure INQ_LIST_OF_FILL_AREA_INDICES
  (WS : in WS_ID;
   ERROR_INDICATOR : out ERROR_NUMBER;
   INDICES : out FILL_AREA_INDICES_LIST_OF);

procedure INQ_FILL_AREA_REPRESENTATION
  (WS : in WS_ID;
   INDEX : in FILL_AREA_INDEX;
   RETURNED_VALUES : in RETURN_VALUE_TYPE;
   ERROR_INDICATOR : out ERROR_NUMBER;
   INTERIOR : out INTERIOR_STYLE;
   STYLE : out STYLE_INDEX;
   FILL_AREA_COLOUR : out COLOUR_INDEX);

procedure INQ_LIST_OF_PATTERN_INDICES
  (WS : in WS_ID;
   ERROR_INDICATOR : out ERROR_NUMBER;
   INDICES : out PATTERN_INDICES_LIST_OF);

procedure INQ_PATTERN_REPRESENTATION
  (WS : in WS_ID;
   INDEX : in PATTERN_INDEX;
   RETURNED_VALUES : in RETURN_VALUE_TYPE;
   ERROR_INDICATOR : out ERROR_NUMBER;
   PATTERN : out VARIABLE_COLOUR_MATRIX);

procedure INQ_LIST_OF_COLOUR_INDICES
  (WS : in WS_ID;
   ERROR_INDICATOR : out ERROR_NUMBER;
   INDICES : out COLOUR_INDICES_LIST_OF);

procedure INQ_COLOUR_REPRESENTATION
  (WS : in WS_ID;
   INDEX : in COLOUR_INDEX;
   RETURNED_VALUES : in RETURN_VALUE_TYPE;
   ERROR_INDICATOR : out ERROR_NUMBER;
   RGB_COLOUR : out COLOUR_REPRESENTATION);

procedure INQ_WS_TRANSFORMATION
  (WS : in WS_ID;
   ERROR_INDICATOR : out ERROR_NUMBER;
   UPDATE : out UPDATE_STATE);

```

```

REQUESTED_WINDOW           : out NDC.RECTANGLE_LIMITS;
CURRENT_WINDOW             : out NDC.RECTANGLE_LIMITS;
REQUESTED_VIEWPORT        : out DC.RECTANGLE_LIMITS;
CURRENT_VIEWPORT          : out DC.RECTANGLE_LIMITS);

procedure INQ_SET_OF_SEGMENT_NAMES_ON_WS
  (WS                       : in WS_ID;
  ERROR_INDICATOR          : out ERROR_NUMBER;
  SEGMENTS                 : out SEGMENT_NAMES_LIST_OF);

procedure INQ_LOCATOR_DEVICE_STATE
  (WS                       : in WS_ID;
  DEVICE                   : in LOCATOR_DEVICE_NUMBER;
  RETURNED_VALUES         : in RETURN_VALUE_TYPE;
  ERROR_INDICATOR          : out ERROR_NUMBER;
  MODE                     : out OPERATING_MODE;
  SWITCH                   : out ECHO_SWITCH;
  INITIAL_TRANSFORMATION   : out TRANSFORMATION_NUMBER;
  INITIAL_POSITION        : out WC_POINT;
  ECHO_AREA                : out DC.RECTANGLE_LIMITS;
  DATA_RECORD            : out LOCATOR_DATA_RECORD);

procedure INQ_STROKE_DEVICE_STATE
  (WS                       : in WS_ID;
  DEVICE                   : in STROKE_DEVICE_NUMBER;
  RETURNED_VALUES         : in RETURN_VALUE_TYPE;
  ERROR_INDICATOR          : out ERROR_NUMBER;
  MODE                     : out OPERATING_MODE;
  SWITCH                   : out ECHO_SWITCH;
  INITIAL_TRANSFORMATION   : out TRANSFORMATION_NUMBER;
  INITIAL_STROKE_POINTS   : out WC_POINT_LIST;
  ECHO_AREA                : out DC.RECTANGLE_LIMITS;
  DATA_RECORD            : out STROKE_DATA_RECORD);

procedure INQ_VALUATOR_DEVICE_STATE
  (WS                       : in WS_ID;
  DEVICE                   : in VALUATOR_DEVICE_NUMBER;
  ERROR_INDICATOR          : out ERROR_NUMBER;
  MODE                     : out OPERATING_MODE;
  SWITCH                   : out ECHO_SWITCH;
  INITIAL_VALUE            : out VALUATOR_INPUT_VALUE;
  ECHO_AREA                : out DC.RECTANGLE_LIMITS;
  DATA_RECORD            : out VALUATOR_DATA_RECORD);

procedure INQ_CHOICE_DEVICE_STATE
  (WS                       : in WS_ID;
  DEVICE                   : in CHOICE_DEVICE_NUMBER;
  ERROR_INDICATOR          : out ERROR_NUMBER;
  MODE                     : out OPERATING_MODE;
  SWITCH                   : out ECHO_SWITCH;
  INITIAL_STATUS           : out CHOICE_STATUS;
  INITIAL_CHOICE           : out CHOICE_VALUE;
  ECHO_AREA                : out DC.RECTANGLE_LIMITS;
  DATA_RECORD            : out CHOICE_DATA_RECORD);

procedure INQ_PICK_DEVICE_STATE
  (WS                       : in WS_ID;
  DEVICE                   : in PICK_DEVICE_NUMBER;

```

```

RETURNED_VALUES      : in RETURN_VALUE_TYPE;
ERROR_INDICATOR      : out ERROR_NUMBER;
MODE                 : out OPERATING_MODE;
SWITCH              : out ECHO_SWITCH;
INITIAL_STATUS      : out PICK_STATUS;
INITIAL_SEGMENT     : out SEGMENT_NAME;
INITIAL_PICK        : out PICK_ID;
ECHO_AREA           : out DC.RECTANGLE_LIMITS;
DATA_RECORD         : out PICK_DATA_RECORD);

procedure INQ_STRING_DEVICE_STATE
  (WS                 : in WS_ID;
   DEVICE             : in STRING_DEVICE_NUMBER;
   ERROR_INDICATOR    : out ERROR_NUMBER;
   MODE               : out OPERATING_MODE;
   SWITCH            : out ECHO_SWITCH;
   INITIAL_STRING     : out INPUT_STRING;
   ECHO_AREA          : out DC.RECTANGLE_LIMITS;
   DATA_RECORD       : out STRING_DATA_RECORD);

procedure INQ_WS_CATEGORY
  (TYPE_OF_WS        : in WS_TYPE;
   ERROR_INDICATOR    : out ERROR_NUMBER;
   CATEGORY           : out WS_CATEGORY);

procedure INQ_WS_CLASSIFICATION
  (TYPE_OF_WS        : in WS_TYPE;
   ERROR_INDICATOR    : out ERROR_NUMBER;
   CLASS             : out DISPLAY_CLASS);

procedure INQ_DISPLAY_SPACE_SIZE
  (TYPE_OF_WS        : in WS_TYPE;
   ERROR_INDICATOR    : out ERROR_NUMBER;
   UNITS             : out DC.UNITS;
   MAX_DC_SIZE       : out DC.SIZE;
   MAX_RASTER_UNIT_SIZE : out RASTER_UNIT_SIZE);

procedure INQ_DYNAMIC_MODIFICATION_OF_WS_ATTRIBUTES
  (TYPE_OF_WS        : in WS_TYPE;
   ERROR_INDICATOR    : out ERROR_NUMBER;
   POLYLINE_REPRESENTATION : out DYNAMIC_MODIFICATION;
   POLYMARKER_REPRESENTATION : out DYNAMIC_MODIFICATION;
   TEXT_REPRESENTATION : out DYNAMIC_MODIFICATION;
   FILL_AREA_REPRESENTATION : out DYNAMIC_MODIFICATION;
   PATTERN_REPRESENTATION : out DYNAMIC_MODIFICATION;
   COLOUR_REPRESENTATION : out DYNAMIC_MODIFICATION;
   TRANSFORMATION     : out DYNAMIC_MODIFICATION);

procedure INQ_DEFAULT_DEFERRAL_STATE_VALUES
  (TYPE_OF_WS        : in WS_TYPE;
   ERROR_INDICATOR    : out ERROR_NUMBER;
   DEFERRAL           : out DEFERRAL_MODE;
   REGENERATION       : out REGENERATION_MODE);

procedure INQ_POLYLINE_FACILITIES
  (TYPE_OF_WS        : in WS_TYPE;

```

```

ERROR_INDICATOR           : out ERROR_NUMBER;
LIST_OF_TYPES              : out LINETYPES.LIST_OF;
NUMBER_OF_WIDTHS          : out NATURAL;
NOMINAL_WIDTH              : out DC.MAGNITUDE;
RANGE_OF_WIDTHS           : out DC.RANGE_OF_MAGNITUDES;
NUMBER_OF_INDICES         : out NATURAL);

procedure INQ_PREDEFINED_POLYLINE_REPRESENTATION
(TYPE_OF_WS                : in WS_TYPE;
INDEX                      : in POLYLINE_INDEX;
ERROR_INDICATOR           : out ERROR_NUMBER;
TYPE_OF_LINE              : out LINETYPE;
WIDTH                     : out LINEWIDTH;
LINE_COLOUR               : out COLOUR_INDEX);

procedure INQ_POLYMARKER_FACILITIES
(TYPE_OF_WS                : in WS_TYPE;
ERROR_INDICATOR           : out ERROR_NUMBER;
LIST_OF_TYPES              : out MARKER_TYPES.LIST_OF;
NUMBER_OF_SIZES           : out NATURAL;
NOMINAL_SIZE              : out DC.MAGNITUDE;
RANGE_OF_SIZES            : out DC.RANGE_OF_MAGNITUDES;
NUMBER_OF_INDICES         : out NATURAL);

procedure INQ_PREDEFINED_POLYMARKER_REPRESENTATION
(TYPE_OF_WS                : in WS_TYPE;
INDEX                      : in POLYMARKER_INDEX;
ERROR_INDICATOR           : out ERROR_NUMBER;
TYPE_OF_MARKER            : out MARKER_TYPE;
SIZE                      : out MARKER_SIZE;
MARKER_COLOUR             : out COLOUR_INDEX);

procedure INQ_TEXT_FACILITIES
(TYPE_OF_WS                : in WS_TYPE;
ERROR_INDICATOR           : out ERROR_NUMBER;
LIST_OF_FONT_PRECISION_  : out
PAIRS                     TEXT_FONT_PRECISION.LIST_OF;
NUMBER_OF_HEIGHTS        : out NATURAL;
RANGE_OF_HEIGHTS         : out DC.RANGE_OF_MAGNITUDES;
NUMBER_OF_EXPANSIONS     : out NATURAL;
EXPANSION_RANGE          : out RANGE_OF_EXPANSIONS;
NUMBER_OF_INDICES         : out NATURAL);

procedure INQ_PREDEFINED_TEXT_REPRESENTATION
(TYPE_OF_WS                : in WS_TYPE;
INDEX                      : in TEXT_INDEX;
ERROR_INDICATOR           : out ERROR_NUMBER;
FONT_PRECISION            : out TEXT_FONT_PRECISION;
EXPANSION                 : out CHAR_EXPANSION;
SPACING                   : out CHAR_SPACING;
TEXT_COLOUR              : out COLOUR_INDEX);

procedure INQ_FILL_AREA_FACILITIES
(TYPE_OF_WS                : in WS_TYPE;
ERROR_INDICATOR           : out ERROR_NUMBER;
LIST_OF_INTERIOR_STYLES  : out INTERIOR_STYLES.LIST_OF;
LIST_OF_HATCH_STYLES     : out HATCH_STYLES.LAST_OF;
NUMBER_OF_INDICES         : out NATURAL);

```

```

procedure INQ_PREDEFINED_FILL_AREA_REPRESENTATION
  (TYPE_OF_WS           : in WS_TYPE;
   INDEX                : in FILL_AREA_INDEX;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   INTERIOR             : out INTERIOR_STYLE;
   STYLE                : out STYLE_INDEX;
   FILL_AREA_COLOUR     : out COLOUR_INDEX);

procedure INQ_PATTERN_FACILITIES
  (TYPE_OF_WS           : in WS_TYPE;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   NUMBER_OF_INDICES    : out NATURAL);

procedure INQ_PREDEFINED_PATTERN_REPRESENTATION
  (TYPE_OF_WS           : in WS_TYPE;
   INDEX                : in PATTERN_INDEX;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   PATTERN              : out VARIABLE_COLOUR_MATRIX);

procedure INQ_COLOUR_FACILITIES
  (TYPE_OF_WS           : in WS_TYPE;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   NUMBER_OF_COLOURS    : out NATURAL;
   AVAILABLE_COLOUR     : out COLOUR_AVAILABLE;
   NUMBER_OF_COLOUR_    : out NATURAL);
INDICES

procedure INQ_PREDEFINED_COLOUR_REPRESENTATION
  (TYPE_OF_WS           : in WS_TYPE;
   INDEX                : in COLOUR_INDEX;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   RGB_COLOUR           : out COLOUR_REPRESENTATION);

procedure INQ_LIST_OF_AVAILABLE_GDP
  (TYPE_OF_WS           : in WS_TYPE;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   LIST_OF_GDP          : out GDP_IDS_LIST_OF);

procedure INQ_GDP
  (TYPE_OF_WS           : in WS_TYPE;
   GDP                  : in GDP_ID;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   LIST_OF_ATTRIBUTES_USED : out ATTRIBUTES_USED_LIST_OF);

procedure ING_MAX_LENGTH_OF_WS_STATE_TABLES
  (TYPE_OF_WS           : in WS_TYPE;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   MAX_POLYLINE_ENTRIES : out NATURAL;
   MAX_POLYMARKER_ENTRIES : out NATURAL;
   MAX_TEXT_ENTRIES     : out NATURAL;
   MAX_FILL_AREA_ENTRIES : out NATURAL;
   MAX_PATTERN_INDICES  : out NATURAL;
   MAX_COLOUR_INDICES   : out NATURAL);

procedure INQ_NUMBER_OF_SEGMENT_PRIORITIES_SUPPORTED
  (TYPE_OF_WS           : in WS_TYPE;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   NUMBER_OF_PRIORITIES : out NATURAL);

```

```

procedure INQ_DYNAMIC_MODIFICATION_OF_SEGMENT_ATTRIBUTES
  (TYPE_OF_WS           : in WS_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   TRANSFORMATION      : out DYNAMIC_MODIFICATION;
   VISIBLE_TO_INVISIBLE : out DYNAMIC_MODIFICATION;
   INVISIBLE_TO_VISIBLE : out DYNAMIC_MODIFICATION;
   HIGHLIGHTING        : out DYNAMIC_MODIFICATION;
   PRIORITY             : out DYNAMIC_MODIFICATION;
   ADDING_PRIMITIVES    : out DYNAMIC_MODIFICATION;
   DELETION_VISIBLE     : out DYNAMIC_MODIFICATION);

procedure INQ_NUMBER_OF_AVAILABLE_LOGICAL_INPUT_DEVICES
  (TYPE_OF_WS           : in WS_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   LOCATOR              : out NATURAL;
   STROKE               : out NATURAL;
   VALUATOR             : out NATURAL;
   CHOICE               : out NATURAL;
   PICK                 : out NATURAL;
   STRING               : out NATURAL);

procedure INQ_DEFAULT_LOCATOR_DEVICE_DATA
  (TYPE_OF_WS           : in WS_TYPE;
   DEVICE               : in LOCATOR_DEVICE_NUMBER;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   INITIAL_POSITION     : out WC_POINT;
   LIST_OF_PROMPT_ECHO_TYPES_ : out LOCATOR_PROMPT_ECHO_TYPES_LIST_OF;
   ECHO_AREA           : out DC_RECTANGLE_LIMITS;
   DATA_RECORD        : out LOCATOR_DATA_RECORD);

procedure INQ_DEFAULT_STROKE_DEVICE_DATA
  (TYPE_OF_WS           : in WS_TYPE;
   DEVICE               : in STROKE_DEVICE_NUMBER;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   MAX_BUFFER_SIZE     : out NATURAL;
   LIST_OF_PROMPT_ECHO_TYPES_ : out STROKE_PROMPT_ECHO_TYPES_LIST_OF;
   ECHO_AREA           : out DC_RECTANGLE_LIMITS;
   DATA_RECORD        : out STROKE_DATA_RECORD);

procedure INQ_DEFAULT_VALUATOR_DEVICE_DATA
  (TYPE_OF_WS           : in WS_TYPE;
   DEVICE               : in VALUATOR_DEVICE_NUMBER;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   INITIAL_VALUE       : out VALUATOR_INPUT_VALUE;
   LIST_OF_PROMPT_ECHO_TYPES_ : out VALUATOR_PROMPT_ECHO_TYPES_LIST_OF;
   ECHO_AREA           : out DC_RECTANGLE_LIMITS;
   DATA_RECORD        : out VALUATOR_DATA_RECORD);

procedure INQ_DEFAULT_CHOICE_DEVICE_DATA
  (TYPE_OF_WS           : in WS_TYPE;
   DEVICE               : in CHOICE_DEVICE_NUMBER;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   MAX_CHOICES         : out CHOICE_VALUE);

```

```

LIST_OF_PROMPT_ECHO_      : out CHOICE_PROMPT_ECHO_
_TYPES_                    : out DC.RECTANGLE_LIMITS;
ECHO_AREA                  : out CHOICE_DATA_RECORD);
DATA_RECORD

procedure INQ_DEFAULT_PICK_DEVICE_DATA
(TYPE_OF_WS                : in WS_TYPE;
DEVICE                      : in PICK_DEVICE_NUMBER;
ERROR_INDICATOR            : out ERROR_NUMBER;
LIST_OF_PROMPT_ECHO_      : out PICK_PROMPT_ECHO_
_TYPES_                    : out DC.RECTANGLE_LIMITS;
ECHO_AREA                  : out PICK_DATA_RECORD);
DATA_RECORD

procedure INQ_DEFAULT_STRING_DEVICE_DATA
(TYPE_OF_WS                : in WS_TYPE;
DEVICE                      : in STRING_DEVICE_NUMBER;
ERROR_INDICATOR            : out ERROR_NUMBER;
MAX_STRING_BUFFER_SIZE    : out NATURAL;
LIST_OF_PROMPT_ECHO_      : out STRING_PROMPT_ECHO_
_TYPES_                    : out DC.RECTANGLE_LIMITS;
ECHO_AREA                  : out STRING_DATA_RECORD);
DATA_RECORD

procedure INQ_SET_OF_ASSOCIATED_WS
(SEGMENT                    : in SEGMENT_NAME;
ERROR_INDICATOR            : out ERROR_NUMBER;
LIST_OF_WS                 : out WS_IDS_LIST_OF);

procedure INQ_SEGMENT_ATTRIBUTES
(SEGMENT                    : in SEGMENT_NAME;
ERROR_INDICATOR            : out ERROR_NUMBER;
TRANSFORMATION             : out TRANSFORMATION_MATRIX;
VISIBILITY                 : out SEGMENT_VISIBILITY;
HIGHLIGHTING               : out SEGMENT_HIGHLIGHTING;
PRIORITY                   : out SEGMENT_PRIORITY;
DETECTABILITY              : out SEGMENT_DETECTABILITY);

procedure INQ_PIXEL_ARRAY_DIMENSIONS
(WC                         : in WS_ID;
CORNER_1_1                 : in WC_POINT;
CORNER_DX_DY              : in WC_POINT;
ERROR_INDICATOR            : out ERROR_NUMBER;
DIMENSIONS                 : out RASTER_UNIT_SIZE);

procedure INQ_PIXEL_ARRAY
(WC                         : in WS_ID;
CORNER                     : in WC_POINT;
DX                         : in RASTER_UNITS;
DY                         : in RASTER_UNITS;
ERROR_INDICATOR            : out ERROR_NUMBER;
INVALID_VALUES             : out INVALID_VALUES_INDICATOR;
PIXEL_ARRAY                : out VARIABLE_PIXEL_COLOUR_
MATRIX);

procedure INQ_PIXEL
(WC                         : in WS_ID;
POINT                      : in WC_POINT;

```

ERROR\_INDICATOR : out ERROR\_NUMBER;  
 PIXEL\_COLOUR : out PIXEL\_COLOUR\_INDEX);

procedure INQ\_INPUT\_QUEUE\_OVERFLOW

(ERROR\_INDICATOR : out ERROR\_NUMBER;  
 WS : out WC\_ID;  
 CLASS : out INPUT\_QUEUE\_CLASS;  
 DEVICE : out EVENT\_OVERFLOW\_DEVICE\_NUMBER);

— ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ

procedure EVALUATE\_TRANSFORMATION\_MATRIX

(FIXED\_POINT : in WC\_POINT;  
 SHIFT\_VECTOR : in WC\_VECTOR;  
 ROTATION\_ANGLE : in RADIANS;  
 SCALE\_FACTORS : in TRANSFORMATION\_FACTOR;  
 TRANSFORMATION : in TRANSFORMATION\_MATRIX);

procedure EVALUATE\_TRANSFORMATION\_MATRIX

(FIXED\_POINT : in NDC\_POINT;  
 SHIFT\_VECTOR : in NDC\_VECTOR;  
 ROTATION\_ANGLE : in RADIANS;  
 SCALE\_FACTORS : in TRANSFORMATION\_FACTOR;  
 TRANSFORMATION : out TRANSFORMATION\_MATRIX);

procedure ACCUMULATE\_TRANSFORMATION\_MATRIX

(SOURCE\_TRANSFORMATION : in TRANSFORMATION\_MATRIX);  
 FIXED\_POINT : in WC\_POINT;  
 SHIFT\_VECTOR : in WC\_VECTOR;  
 ROTATION\_ANGLE : in RADIANS;  
 SCALE\_FACTORS : in TRANSFORMATION\_FACTOR;  
 RESULT\_TRANSFORMATION : out TRANSFORMATION\_MATRIX);

procedure ACCUMULATE\_TRANSFORMATION\_MATRIX

(SOURCE\_TRANSFORMATION : in TRANSFORMATION\_MATRIX);  
 FIXED\_POINT : in NDC\_POINT;  
 SHIFT\_VECTOR : in NDC\_VECTOR;  
 ROTATION\_ANGLE : in RADIANS;  
 SCALE\_FACTORS : in TRANSFORMATION\_FACTOR;  
 RESULT\_TRANSFORMATION : out TRANSFORMATION\_MATRIX);

— ФУНКЦИИ ОБРАБОТКИ ОШИБОК

procedure ERROR\_LOGGING

(ERROR\_INDICATOR : in ERROR\_NUMBER;  
 GKS\_FUNCTION : in STRING;  
 ERROR\_FILE : in STRING := DEFAULT\_ERROR\_FILE);

procedure EMERGENCY\_CLOSE\_GKS;

— Утилиты функций метафайла

— Элементы записей метафайла могут содержать списки указателей, строки символов, матрицы индексов цветов, данные GDP и ESC. Длина записи зависит от числа элементов данных. ЯГС определяет, что формат зависит от реализации.

— Тип элемента записи данных должен быть личным, позволяющим манипулировать содержимым записей, чтобы повысить эффективность их обработки.

— Прикладной программист должен быть способен записывать не графические данные в метафайл. Этого можно достигнуть, разрешив вывод литер. Числен-

ные данные должны быть преобразованы в строки прикладным программистом до вызова процедуры BUILD\_NEW\_GKSM\_DATA\_RECORD.

```

procedure BUILD_NEW_GKSM_DATA_RECORD
  (TYPE_OF_ITEM           : in GKSM_ITEM_TYPE;
   ITEM_DATA             : in STRING;
   ITEM                  : out GKSM_DATA_RECORD);

function ITEM_DATA_RECORD_STRING
  (ITEM : in GKSM_DATA_RECORD) return STRING;
private
— Следующие типы определяют спецификации для личных записей данных.
type GKSM_DATA_RECORD           : GKSM_ITEM_TYPE := 0;
  (TYPE_OF_ITEM
   LENGTH : NATURAL := 0) is
  record
    null;
  end record;

type CHOICE_DATA_RECORD (PROMPT_ECHO_TYPE:
  CHOICE_PROMPT_ECHO_TYPE := DEFAULT_CHOICE) is
  record
    null;
  end record;

type LOCATOR_DATA_RECORD (PROMPT_ECHO_TYPE:
  LOCATOR_PROMPT_ECHO_TYPE := DEFAULT_LOCATOR) is
  record
    null;
  end record;

type STRING_DATA_RECORD (PROMPT_ECHO_TYPE:
  STRING_PROMPT_ECHO_TYPE := DEFAULT_STRING) is
  record
    null;
  end record;

type STROKE_DATA_RECORD (PROMPT_ECHO_TYPE:
  STROKE_PROMPT_ECHO_TYPE := DEFAULT_STROKE) is
  record
    null;
  end record;

type VALUATOR_DATA_RECORD (PROMPT_ECHO_TYPE:
  VALUATOR_PROMPT_ECHO_TYPE := DEFAULT_VALUATOR) is
  record
    null;
  end record;

type PICK_DATA_RECORD (PROMPT_ECHO_TYPE:
  PICK_PROMPT_ECHO_TYPE := DEFAULT_PICK) is
  record
    null;
  end record;
end GKS;

```

— Функция обработки ошибок является отдельным библиотечным блоком и не компилируется как часть пакета ЯГС.

```
procedure ERROR_HANDLING
  (ERROR_INDICATOR      : in ERROR_NUMBER;
   GKS_FUNCTION         : in STRING;
   ERROR_FILE           : in STRING := DEFAULT_ERROR_
                                     FILE);
```

```
with GKS_TYPES;
use GKS_TYPES;
package GKS_GDP is
```

— Пакет ОПВ является отдельным библиотечным блоком и не компилируется как часть ЯГС.

ОПВ связывается с отдельными процедурами, реализованными для каждого обобщенного примитива вывода, каждая со своим собственным интерфейсом. Имена ОПВ и параметры регистрируют в Международном журнале графических записей ИСО, который ведется органом регистрации.

Каждая незарегистрированная процедура ОПВ, поддерживаемая реализацией, будут содержаться в отдельном библиотечном пакете. При этом используются следующие соглашения по наименованию:

```
package GKS_UGDP_< имя процедуры ОПВ > is
  procedure GDP;
— Код на языке Ада для процедуры ОПВ;
— Единственным именем процедуры, используемым в пакете, будет ОПВ.
— Для того, чтобы поддержать возможность записывать не реализованные ОПВ
в метафайл, могут быть привлечены зарегистрированные ОПВ в форме процедуры
GENERALIZED_GDP, которая имеет спецификацию, показанную ниже:
type GDP_FLOAT is digits PRECISION;
type GDP_INTEGER_ARRAY is array (SMALL_NATURAL range <>)
  of INTEGER;
type GDP_FLOAT_ARRAY is array (SMALL_NATURAL range <>)
  of GDP_FLOAT;
type GDP_STRING_ARRAY is array (SMALL_NATURAL range <>)
  of STRING (1..80);
type GDP_DATA_RECORD (NUM_OF_INTEGERS : SMALL_NATURAL := 0;
  NUM_OF_REALS      : SMALL_NATURAL := 0;
  NUM_OF_STRINGS   : SMALL_NATURAL := 0) is
record
  INTEGER_ARRAY : GDP_INTEGER_ARRAY (1..NUM_OF_INTEGERS);
  REAL_ARRAY   : GDP_FLOAT_ARRAY (1..NUM_OF_REALS);
  GDP_STRINGS  : GDP_STRING_ARRAY (1..NUM_OF_STRINGS);
end record,
procedure GENERALIZED_GDP      : in GDP_ID;
  (GDP_NAME                     POINT : in WC_POINT_LIST;
   GDP_DATA                     : out GDP_DATA_RECORD);
end GKS_GDP;
with GKS_TYPES;
use GKS_TUPES;
package GKS_ESCAPE is
```

— Пакет ESCAPE является отдельным библиотечным блоком и не компилируется как часть ЯГС.

— Функции расширения связываются в Ада как отдельные процедуры для каждого уникального типа расширения, предоставляемого реализацией, каждая со списком формальных параметров, соответствующим реализованной процедуре. Имена ESCAPE и параметры регистрируют в Международном журнале графических записей ИСО, который ведется органом регистрации.

— Каждая незарегистрированная процедура ESCAPE будет отдельным библиотечным пакетом, использующим следующие соглашения по наименованию:

```
package GKS_UESC : <имя процедуры ESCAPE> is
```

```
  procedure ESC;
```

```
  — Код на языке Ада процедуры UESC;
```

```
end GKS_UESC <имя процедуры ESCAPE>;
```

— Единственным именем процедуры, используемым в пакете, является ESC.

— Для того чтобы поддержать возможность записывать не реализованные ESCAPE в метафайл могут быть привлечены зарегистрированные процедуры ESCAPES в форме процедуры GENERALIZED\_ESC, которая имеет спецификации, показанные ниже:

```
type ESCAPE_ID in new INTEGER;
```

```
type ESCAPE_FLOAT is digits PRECISION;
```

```
type ESC_INTEGER_ARRAY is array (SMALL_NATURAL range <>)
  of INTEGER;
```

```
type ESC_FLOAT_ARRAY is array (SMALL_NATURAL range <>)
  of ESCAPE_FLOAT;
```

```
type ESC_STRING_ARRAY is array (SMALL_NATURAL range <>)
  of STRING (1..80);
```

```
type ESC_DATA_RECORD : SMALL_NATURAL := 0;
```

```
  (NUM_OF_INTEGERS
```

```
    NUM_OF_REALS : SMALL_NATURAL := 0;
```

```
    NUM_OF_STRINGS : SMALL_NATURAL := 0) is
```

```
record
  INTEGER_ARRAY : ESC_INTEGER_ARRAY (1..NUM_OF_INTEGERS);
```

```
  REAL_ARRAY : ESC_FLOAT_ARRAY (1..NUM_OF_REALS);
```

```
  ESC_STRING : ESC_STRING_ARRAY (1..NUM_OF_STRINGS);
```

```
end record;
```

```
procedure GENERALIZED_ESC : in ESCAPE_ID;
```

```
  (ESCAPE_NAM
```

```
    ESC_DATA_IN : in ESC_DATA_RECORD;
```

```
    ESC_DATA_OUT : out ESC_DATA_RECORD;
```

```
end GKS_ESCAPE;
```

## СПИСОК ССЫЛОК НА ОПРЕДЕЛЕННЫЕ РЕАЛИЗАЦИИ ЗАПИСИ

(Это приложение не является составной частью стандарта, но дает дополнительную информацию)

Элемент	Пункт
CHOICE_DATA_RECORD	4 2 3
LOCATOR_DATA_RECORD	4 2 3
PICK_DATA_RECORD	4 2 3
STRING_DATA_RECORD	4 2 3
STROKE_DATA_RECORD	4 2 3
VALUATOR_DATA_RECORD	4 2 3
DEFAULT_MEMORY_UNITS	4 2 4
DEFAULT_ERROR_FILE	4 2 4
PRECISION	4 2 4
MAX_LIST_SIZE	5 2 3
SMALL_NATURAL_MAX	4 2 4
CHOICE_SMALL_NATURAL_MAX	4 2 4
STRING_SMALL_NATURAL_MAX	4 2 4

## ПРИМЕРЫ ПРОГРАММ

(Это приложение не является составной частью стандарта, но дает дополнительную информацию).

В настоящем приложении даны программы, использующие связь, определенную в стандарте

## В 1. Пример программы 1: STAR

## — ПРОГРАММА STAR

## — ОПИСАНИЕ:

— Эта программа рисует желтую звезду на голубом фоне и пишет зеленым цветом STAR под звездой.

## — СОГЛАШЕНИЯ:

— Уровень ЯГС 0a

— Реализация должна поддерживать, по крайней мере, работу одной станции категории вывода или ввода/вывода.

with GKS;

with GKS\_TYPES;

use GKS;

use GKS\_TYPES;

procedure STAR is

— Определяет переменные станции и файл регистрации ошибок.

```

MY_WS_ID           : constant WS_ID           := 1;
SOME_CONNECTION    : constant STRING         := «UNIT_1»;
SOME_OUTPUT_TYPE   : constant WS_TYPE        := 1;
ERROR_FILE         : constant STRING         := «MY_ERROR_FILE»;
  
```

— Определяет точки звезды.

```

STAR_POINTS : constant WS_POINT_ARRAY :=
  ((0.951057, 0.309017),
  (-0.951057, 0.309017),
  (0.587785, -0.951057),
  (0.0, 1.0),
  (-0.587785, -0.951057));
  
```

— Определяет окно мировой системы и различные атрибуты.

```

WINDOW:WC.RECTANGLE_LIMITS:=
  (XMIN=>-1.25, XMAX=>1.25,
  YMIN=>-1.25, YMAX=>1.25);
  
```

```

TEXT_POSITION.WC.POINT:= (0.0, -1.0);
begin
  
```

begin

— Открыть ЯГС и активизировать станцию.

```

OPEN_GKS (ERROR_FILE);
  
```

```

OPEN_WS (MY_WS_ID SOME_CONNECTION.SOME_OUTPUT_TYPE);
  
```

```

ACTIVATE_WS (MY_WS_ID);
  
```

— Центрировать окно вокруг начала координат.

```
SET WINDOW (1, WINDOW);
SELECT_NORMALIZATION_TRANSFORMATION (1);
```

— Задать цвета.

```
SET_COLOUR_REPRESENTATION (MY_WS_ID,
                             INDEX=>0,
                             RGB_COLOUR=>(0.0, 0.0, 1.0));
SET_COLOUR_REPRESENTATION (MY_WS_ID,
                             INDEX=>1,
                             RGB_COLOUR=>(1.0, 1.0, 0.0));
SET_COLOUR_REPRESENTATION (MY_WS_ID,
                             INDEX=>2,
                             RGB_COLOUR=>(1.0, 1.0, 1.0));
```

— Установить атрибуты области заполнения.

```
SET_FILE_AREA_INTERIOR_STYLE (SOLID);
SET_FILE_AREA_COLOUR_INDEX (1);
```

— Нарисовать звезду.

```
FILL_AREA (STAR_POINTS);
```

— Выбрать заглавные буквы, позиционированные под звездой.

```
SET_CHAR_HEIGHT (HEIGHT=>0 15);
SET_TEXT_ALIGNMENT (ALIGNMENT=>(CENTRE, HALF));
SET_TEXT_COLOUR_INDEX (TEXT_COLOUR=>2);
```

— Нарисовать заголовок.

```
TEXT (TEXT_POSITION, «STAR»);
```

— Закрыть станцию и ЯГС.

```
DEACTIVATE_WS (MY_WS_ID);
CLOSE_WS (MY_WS_ID);
CLOSE_GKS;
```

end STAR;

## В.2. Пример программы 2: IRON

— ПРОГРАММА IRON

— ОПИСАНИЕ:

— Данная программа рисует горизонтальную гистограмму, иллюстрирующую цены в металлургической промышленности.

— Пользователь может выбрать данные для отображения, используя устройство выбора альтернативы ЯГС. График адаптирован из журнала «Scientific American», май 1984 г., стр. 139.

— СОГЛАШЕНИЕ:

— Уровень ЯГС 2b

with GKS;

with GKS\_TYPES;

use GKS;

use GKS\_TYPES;

procedure IRON is

— Задает переменные станции и файл регистрации ошибок.

```
MY_WS_ID : constant WS_ID := 1;
```

SOME\_CONNECTION : constant STRING := «TTY»;  
 SOME\_OUTIN\_TYPE : constant WS\_TYPE := 2;  
 ERROR\_FILE : constant STRING := «MY\_ERROR\_FILE»;

— Описать и инициализировать флаги выборки атрибутов (используют связи для области заголовка, в противном случае устанавливают индивидуальный).

ASF\_SETTINGS:ASF\_LIST :=  
 (TYPE\_OF\_LINE\_ASF :=> INDIVIDUAL,  
 WIDTH\_ASF :=> INDIVIDUAL,  
 LINE\_COLOUR\_ASF :=> INDIVIDUAL,  
 TYPE\_OF\_MARKER\_ASF :=> INDIVIDUAL,  
 SIZE\_ASF :=> INDIVIDUAL,  
 MARKER\_COLOUR\_ASF :=> INDIVIDUAL,  
 FONT\_PRECISION\_ASF :=> INDIVIDUAL,  
 EXPANSION\_ASF :=> INDIVIDUAL,  
 SPACING\_ASF :=> INDIVIDUAL,  
 TEXT\_COLOUR\_ASF :=> INDIVIDUAL,  
 INTERIOR\_ASF :=> BUNDLED,  
 STYLE\_ASF :=> BUNDLED,  
 FILL\_AREA\_COLOUR\_ASF :=> INDIVIDUAL);

— Описать и инициализировать объекты для устройства выбора альтернатив.

CHOICE\_STRING\_COUNT : constant := 3;  
 CHOICE\_DEVICE : constant CHOICE\_DEVICE\_NUMBER := 1;  
 CHOICE\_ERROR : ERROR\_NUMBER;  
 CHOICE\_MODE : OPERATING\_MODE;  
 CHOICE\_ECHO\_SWITCH : ECHO\_SWITCH;  
 INITIAL\_CHOICE : CHOICE\_VALUE;  
 CHOICE\_INPUT\_RECORD : CHOICE\_DATE\_RECORD;  
 CHOICE\_ECHO\_AREA : DC.RECTANGLE\_LIMITS;  
 PROMPT\_ECHO\_TYPE : constant CHOICE\_PROMPT\_ECHO\_TYPE := 3;  
 CHOICE\_STRINGS : constant CHOICE\_PROMPT\_STRING\_LIST := (LENGTH => 3,  
 LIST => ((4, «U. S.»), (9, «W.GERMANY»), (5, «JAPAN»)));  
 CHOICE\_RECORD : CHOICE\_DATA\_RECORD;  
 CHOICE : CHOICE\_VALUE;  
 CHOICE\_REQUEST : CHOICE\_REQUEST\_STATUS;  
 INITIAL\_STATUS : CHOICE\_STATUS;

— Описать объекты цвета.

RGB\_COLOUR : COLOUR\_REPRESENTATION;  
 WHITE : constant COLOUR\_INDEX := 0;  
 BLACK : constant COLOUR\_INDEX := 1;  
 RED : constant COLOUR\_INDEX := 2;

— Описать и инициализировать объекты окна.

WINDOW\_1 : TRANSFORMATION\_NUMBER := 1;  
 WINDOW\_LIMITS : WC.RECTANGLE\_LIMITS :=  
 (XMIN => -100.0, XMAX => 175.0,  
 YMIN => -2.0, YMAX => 13.0);

— Описать и инициализировать объекты данных гистограммы

MAX\_DATA : constant := 6;

```

type IRON_DATA is array (1..MAX_DATA) of WS_TYPE;
US_DATA_1          : IRON_DATA := (69.0, 50.0, 15.0,
                                   53.0, 57.0, 150.0);
US_DATA_2          : IRON_DATA := (72.0, 50.0, 103.0, 0.0,
                                   0.0, 56.0);
GERMANY_DATA_1    : IRON_DATA := (65.0, 42.0, 3.0, 89.0,
                                   52.0, 93.0);
GERMANY_DATA_2    : IRON_DATA := (70.0, 53.0, 102.0, 0.0,
                                   0.0, 49.0);
JAPAN_DATA_1      : IRON_DATA := (65.0, 47.0, 2.0, 60.0,
                                   52.0, 55.0);
JAPAN_DATA_2      : IRON_DATA := (70.0, 57.0, 105.0, 0.0,
                                   0.0, 41.0);

procedure BARS (LENGTH)   - WC_TYPE;
                    POSITION : WC_POINT) is
LEFT_HALF              : TEXT_ALIGNMENT := (LEFT,
                                           HALF);
BAR_POINTS              : WC_POINT_ARRAY(1..4);

begin if LENGTH = 0.0 then
    SET_TEXT_ALIGNMENT (LEFT_HALF);
    TEXT (POSITION, «0»);
else
    BAR_POINTS := ((X => 0.0,   Y => POSITION.Y+0.4),
                  (X => LENGTH, Y => POSITION.Y+0.4),
                  (X => LENGTH, Y => POSITION.Y-0.4),
                  (X => 0.0,   Y => POSITION.Y-0.4));

    FILL_AREA (BAR_POINTS),
end if;
end BARS;

procedure TICKS (TICK_MARK_POSITION : in out WC_POINT_ARRAY) is
TICK_MARK_LABEL_POSITION : WC_POINT;
begin
for I in 1..4 loop
    POLYLINE (TICK_MARK_POSITION);
    TICK_MARK_POSITION (1).X := TICK_MARK_POSITION (1).X + 50.0;
    TICK_MARK_POSITION (2).X := TICK_MARK_POSITION (2).X + 50.0;
end loop;

-- Нарисовать маскировочные метки.
TICK_MARK_LABEL_POSITION.X := 0.0;
TICK_MARK_LABEL_POSITION.Y := WC_TYPE
(TICK_MARK_POSITION (1).Y);
TEXT (TICK_MARK_LABEL_POSITION, «0»);
TICK_MARK_LABEL_POSITION.X := 50.0;
TEXT (TICK_MARK_LABEL_POSITION, «50»);
TICK_MARK_LABEL_POSITION.X := 100.0;
TEXT (TICK_MARK_LABEL_POSITION, «100»);
TICK_MARK_LABEL_POSITION.X := 150.0;
TEXT (TICK_MARK_LABEL_POSITION, «150»);
end TICKS;

procedure BORDER is

```

— Нарисовать границу, окружающую данные.

```

LABEL_POSITION      WC.POINT;
TITLE_POSITION      WC.POINT = (37.5, -2.0);
HEIGHT              WC.MAGNITUDE = 0.5;
LEFT_HALF           TEXT_ALIGNMENT := (LEFT,
                                     HALF);
CENTRE_BOTTOM       : TEXT_ALIGNMENT := (CENTRE,
                                     BOTTOM);
CENTRE_CAP           : TEXT_ALIGNMENT := (CENTRE,
                                     CAP);
ONLY_IF_NOT_EMPTY   CONTROL_FLAG
                    := CONDITIONALLY;
BOX_POINTS           : constant WC.POINT_ARRAY :=
                    ((0.0, 0.0), (150.0, 0.0), (150.0, 12.0), (0.0, 12.0),
                    (0.0, 0.0));

```

type LABELS is array (1..6) of INPUT\_STRING;

```

BAR_LABELS : constant LABELS :=
    ((5, «LABOR»), (8, «IRON ORE»), (12, «COKE OR COAL»),
     (15, «PURCHASED SCRAP»), (11, «OTHER COSTS»),
     (12, «OTHER ENERGY»));

```

```

TOP_TICK_MARK_START : WC.POINT_ARRAY (1..2) := ((0.0, 12.0),
                                                  (0.0, 11.9));
BOTTOM_TICK_MARK_START : WC.POINT_ARRAY (1..2) := ((0.0, 0.0),
                                                    (0.0, 0.1));

```

begin

```
CLEAR_WS (MY_WS_ID, ONLY_IF_NOT_EMPTY);
```

— Нарисовать квадрат, ограниченный областью диаграммы.

```
POLYLINE (BOX_POINTS);
```

— Нарисовать метки гистограммы, центрированные по полоскам.

```
SET_TEXT_ALIGNMENT (LEFT_HALF);
```

```
SET_CHAR_HEIGHT (HEIGHT);
```

```
SET_TEXT_COLOUR_INDEX (BLACK);
```

```
LABEL_POSITION X := -99.0;
```

```
for I in 1..6 loop
```

```
    LABEL_POSITION.Y := WC_TYPE (2.0 * (FLOAT(I) - 1.0) + 1.2)
```

```
    TEXT (LABEL_POSITION, BAR_LABELS (INTEGER(I)).CONTENTS);
```

```
end loop;
```

— Нарисовать верхние и нижние черточки (красное основание).

```
SET_TEXT_ALIGNMENT (CENTRE_BOTTOM);
```

— Вызвать процедуры для рисования черточек.

```
TICKS (TOP_TICK_MARK_START);
```

```
SET_TEXT_ALIGNMENT (CENTRE_CAP);
```

```
SET_TEXT_COLOUR_INDEX (RED);
```

```
TICKS (BOTTOM_TICK_MARK_START);
```

— Нарисовать заголовок.

```
SET_TEXT_COLOUR_INDEX (BLACK);
```

```
SET_TEXT_ALIGNMENT (CENTRE_BOTTOM);
```

```
TEXT (TITLE_POSITION, «PRODUCTION COST»);
```

```
end BORDER;
```

```

procedure DRAW
    (DATA1 : in out IRON_DATA;
     DATA2 : in out IRON_DATA) is
    : FILL_AREA_INDEX := 1;
    : WC.POINT;
begin
    — Нарисовать границу.
    BORDER;

    — Нарисовать черные полосы.
    SET_FILL_AREA_COLOUR_INDEX (BLACK);
    SET_TEXT_COLOUR_INDEX (BLACK);
    SET_FILL_AREA_INDEX (FILL_INDEX);
    for I in 1..6 loop
        POSITION.Y := 2.0*(WC_TYPE(I)-1.0)+1.6;
        — Вызвать процедуру, которая вычеркивает гистограммы.
        BARS(DATA1(INTEGER(I)), POSITION);
    end loop;

    — Вычеркивает красные гистограммы.
    SET_FILL_AREA_COLOUR_INDEX (RED);
    SET_TEXT_COLOUR_INDEX (RED);
    FILL_INDEX := 2;

    SET_FILL_AREA_INDEX (FILL_INDEX);
    for I in 1..6 loop
        POSITION.Y := 2.0*(WS_TYPE(I)-1.0)+1.6;
        — Вызвать процедуру, которая вычеркивает гистограммы.
        BARS(DATA2(INTEGER(I)), POSITION);
    end loop;
end DRAW;
begin
    — Открыть ЯГС и активировать станцию.
    OPEN_GKS(ERROR_FILE);
    OPEN_WS(MY_WS_ID, SOME_CONNECTION.SOME_OUTIN_TYPE);
    ACTIVATE_WS(MY_WS_ID);

    — Задать окно на диаграмме.
    SET_WINDOW (WINDOW_1, WINDOW_LIMITS);
    SELECT_NORMALIZATION_TRANSFORMATION (WINDOW_1);

    — Задать цвета, которые будут использоваться
    SET_COLOUR_REPRESENTATION (MY_WS_ID,
                                INDEX => WHITE,
                                RGB_COLOUR => (1.0, 1.0, 1.0));
    SET_COLOUR_REPRESENTATION (MY_WS_ID,
                                INDEX => BLACK,
                                RGB_COLOUR => (0.0, 0.0, 0.0));
    SET_COLOUR_REPRESENTATION (MY_WS_ID,
                                INDEX => RED,
                                RGB_COLOUR => (1.0, 0.0, 0.0));

    — Использовать связанные атрибуты, исключая цвет
    SET ASF (ASF_SETTINGS);

```

- Инициализировать устройство выбора  
**INQ\_CHOICE\_DEVICE\_STATE** (MY\_WS\_ID,  
 CHOICE\_DEVICE,  
 CHOICE\_ERROR,  
 CHOICE\_MODE,  
 CHOICE\_ECHO\_SWITCH,  
 INITIAL\_STATUS,  
 INITIAL\_CHOICE,  
 CHOICE\_ECHO\_AREA,  
 CHOICE\_RECORD);
- BUILD\_CHOICE\_DATA\_RECORD** (PROMPT\_ECHO\_TYPE,  
 CHOICE\_STRINGS,  
 CHOICE\_RECORD);
- INITIALISE\_CHOICE** (MY\_WS\_ID,  
 CHOICE\_DEVICE,  
 INITIAL\_STATUS,  
 INITIAL\_CHOICE,  
 CHOICE\_ECHO\_AREA,  
 CHOICE\_RECORD);
- Получить выбор пользователя (U S., W. GERMANY, or JAPAN).  
 loop  
**REQUEST\_CHOICE** (MY\_WS\_ID,  
 CHOICE\_DEVICE,  
 INITIAL\_STATUS,  
 CHOICE);  
 if INITIAL\_STATUS=OK then  
   case CHOICE is  
     when 1 => DRAW (US\_DATA\_1, US\_DATA\_2);  
     when 2 => DRAW (GERMANY\_DATA\_1, GERMANY\_DATA\_2);  
     when 3 => DRAW (JAPAN\_DATA\_1, JAPAN\_DATA\_2);  
     when others => exit,  
   end case;  
 else  
   exit;  
 end if;  
 end loop;
- Закрыть станцию и ЯГС.  
**DEACTIVATE\_WS** (MY\_WS\_ID),  
**CLOSE\_WS** (MY\_WS\_ID),  
**CLOSE\_GKS**,  
 end IRON;

### В. 3. Пример программы 3: MAP

#### — ПРОГРАММА MAP

#### — ОПИСАНИЕ:

— Эта программа считывает метафайл ЯГС для вычерчивания карты. Примитивы каждой области заключены в отдельных сегментах. Пользователь может применить для указания области устройство выбора. Опрашиваемое устройство выбора задает операции с выбранными примитивами области.

#### — СОГЛАШЕНИЕ:

— Уровень ЯГС 1c

— Реализация должна поддерживать, по крайней мере, одну станцию категории ввод/вывод и одну категорию метафайла ввода. Устройство выбора по умолчанию должно поддерживать, по крайней мере, пять альтернатив.

with GKS;

with GKS\_TYPE;

use GKS;

use GKS\_TYPES;

procedure METAFILE is

— Задать станцию метафайла.

```

METAFILE_WS_ID          : constant WS_ID := 1;
METAFILE_CONNECTION     : constant STRING := «METAFILE_
                        _INPUT_FILE»;
METAFILE_TYPE           : constant WS_TYPE := 2;
METAFILE_ITEM_TYPE      : constant GKSM_ITEM_TYPE;
METAFILE_DATA_RECORD    : constant GKSM_DATA_RECORD;
LENGTH, MAX_LENGTH      : NATURAL := 500;

```

— Задать станцию ввода/вывода.

```

MY_WS_ID                : constant WS_ID := 2;
SOME_CONNECTION         : constant STRING := «UNIT_2»;
SOME_OUTIN_TYPE        : constant WS_TYPE := 1;
SOME_CHOICE_DEVICE     : constant CHOICE_DEVICE_
                        _NUMBER := 1;
CSTATUS                 : CHOICE_STATUS;
CHOICE_NUMBER          : CHOICE_VALUE;
SOME_PICK_DEVICE       : constant PICK_DEVICE_
                        _NUMBER := 1;
PSTATUS                 : PICK_REQUEST_STATUS;
PICK                    : PICK_ID;
SEGMENT                 : SEGMENT_NAME;

```

— Задать файл регистрации ошибок.

```

ERROR_FILE              : constant STRING := «MY_ERROR_
                        _FILE»;

```

begin

— Открыть ЯГС и активизировать станции.

```

OPEN_GKS(ERROR_FILE);
OPEN_WS(METAFILE_WS_ID, METAFILE_CONNECTION, METAFILE_TYPE);
OPEN_WS(MY_WS_ID, SOME_CONNECTION, SOME_OUTIN_TYPE);
ACTIVATE_WS(MY_WS_ID);

```

— Установить устройство выбора в режим выборки.

```

SET_CHOICE_MODE (MY_WS_ID, SOME_CHOICE_DEVICE,
                SAMPLE_MODE, NOECHO);

```

— Интерпретировать элементы метафайла до считывания конца метафайла.

```

loop
  GET_ITEM_TYPE_FROM_GKSM (METAFILE_WS_ID,
                          METAFILE_ITEM_TYPE, LENGTH);
  if METAFILE_ITEM_TYPE=0 then
    exit;
  end if;
  READ_ITEM_FROM_GKSM (METAFILE_WS_ID, MAX_LENGTH,
                      METAFILE_DATA_RECORD);

```

```

INTERPRENT_ITEM (METAFILE_DATA_RECORD);
end loop;
— Закрыть станцию метафайла.
CLOSE_WS (METAFILE_WS_ID);
— Позволить пользователю выбрать состояния до выбора EXIT.
loop
REQUEST_PICK
(MY_WS_ID, SOME_PICK_DEVICE, PSTATUS, SEGMENT, PICK);
if PSTATUS=OK then
SAMPLE_CHOICE (MY_WS_ID, SOME_CHOICE_DEVICE, CSTATUS,
CHOICE_NUMBER);
if CSTATUS=OK then
case CHOICE_NUMBER is
when 1=>SET_HIGHLIGHTING (SEGMENT, HIGHLIGHTED);
when 2=>SET_HIGHLIGHTING (SEGMENT, NORMAL);
when 3=>SET_VISIBILITY (SEGMENT, INVISIBLE);
when 4=>SET_VISIBILITY (SEGMENT, VICIBLE)
when others=>null;
end case;
end if;
end if;
end loop;
— Закрыть станцию и ЯГС.
DEACTIVATE_WS (MY_WS_ID);
CLOSE_WS (MY_WS_ID);
CLOSE_GKS;
end METAFILE;

```

#### В. 4. Пример программы 4: MANIPULATE

— ПРОГРАММА MANIPULATE

— ОПИСАНИЕ:

— Данная программа позволяет пользователю создать объект и затем манипулировать им, изменяя преобразование сегмента.

— СОГЛАШЕНИЯ:

— Уровень ЯГС: 2b

with GKS;

with GKS\_TYPES;

use GKS;

use GKS\_TYPES;

procedure POLYGON is

POINTS	: WC.POINT_ARRAY(1..500);
POINT_1	: WC.POINT := (0.6, 0.4);
POINT_2	: WC.POINT := (0.4, 0.3);
POLYGON_SEGMENT	: SEGMENT_NAME := 1;
SHIFT	: constant CHOICE_VALUE := 1;
ZOOM	: constant CHOICE_VALUE := 2;
ROTATE	: constant CHOICE_VALUE := 3;
NEXT	: POSITIVE := 1
TRANSFORMATION	: TRANSFORMATION_NUMBER;
TRANSFORMATION_1	: TRANSFORMATION_NUMBER;
TRANSFORMATION_2	: TRANSFORMATION_NUMBER;

```

RED : constant COLOUR_INDEX :=2;
AXIC_CHARACTER_HEIGHT : constant WC.MAGNITUDE :=0.02;
CHOICE : CHOICE_VALUE;
CHOICE_STATUS : CHOICE_REQUEST_STATUS;
LOCATOR_STATUS : INPUT_STATUS;
MATRIX : TRANSFORMATION_MATRIX;
MATRIX_RESULT : TRANSFORMATION_MATRIX;

```

— Задать переменные станции и файл регистрации ошибок.

```

DISPLAY : constant WS_ID :=1;
DISPLAY_CONNECTION : constant STRING :=«DDDIS»;
DISPLAY_TYPE : constant WS_TYPE :=3;
ERROR_FILE : constant STRING :=«MY_ERROR_FILE»;

```

— Задать переменные сегмента станции.

```

SEGSTORE : constant WS_ID :=2;
SEG_CONNECTION : constant STRING :=«DDSEG»;
SEG_TYPE : constant WS_TYPE :=4;

```

— Задать переменные станции графопостроителя.

```

PLOTTER : constant WS_ID :=6;
PLOT_CONNECTION : constant STRING :=«PLOT»;
PLOT_TYPE : constant WS_TYPE :=5;

```

— Задайте окно мировой системы координат и другие атрибуты.

```

WINDOW_BOUNDS : WC.RECTANGLE_LIMITS :=
    (XMIN=>0.0, XMAX=>1.0,
     YMIN=>0.0, YMAX=>1.0);
VIEWPORT_BOUNDS : NDC.RECTANGLE_LIMITS :=
    (XMIN=>0.0, XMAX=>1.0,
     YMIN=>0.0, YMAX=>1.0);
TEXT_POSITION : WC.POINT := (0.5, 0.5);
begin

```

— Открыть ЯГС и активировать станцию.

```

OPEN_GKS (ERROR_FILE);
OPEN_WS (DISPLAY, DISPLAY_CONNECTION, DISPLAY_TYPE);
ACTIVATE_WS (DISPLAY);
OPEN_WS (SEGSTORE, SEG_CONNECTION, SEG_TYPE);
ACTIVATE_WS (SEGSTORE);
SET_WINDOW (1, WINDOW_BOUNDS);
SET_VIEWPORT (1, VIEWPORT_BOUNDS);
SET_VIEWPORT_INPUT_PRIORITY (1, 0, HIGHER);

```

— Построение сегмента POLYGON\_SEGMENT

```

CREATE_SEGMENT (POLYGON_SEGMENT);
SET_POLYLINE_INDEX (3);
REQUEST_LOCATOR (DISPLAY, LOCATOR_STATUS,
    TRANSFORMATION, POINTS (NEXT));
SELECT_NORMALIZATION_TRANSFORMATION (TRANSFORMATION, -1);
loop

```

```

    NEXT:=NEXT+1;
    REQUEST_LOCATOR (DISPLAY, 1, LOCATOR_STATUS,
    TRANSFORMATION, POINTS (NEXT));

```

```

exit when LOCATOR_STATUS = NONE or
      TRANSFORMATION /= TRANSFORMATION_1 or
      NEXT = 500;
end loop;
POINTS (NEXT) := POINTS(1);
POLYLINE(POINTS);
CLOSE_SEGMENT;
EVALUATE_TRANSFORMATION_MATRIX (WC.POINT((0.0, 0.0)),
      WC.VECTOR' ((0.0, 0.0)), 0.0, (1.0, 1.0), MATRIX;
— инициализировать матрицу преобразования
loop
  REQUEST_CHOICE (DISPLAY, 1, CHOICE_STATUS, CHOICE);
  exit when CHOICE_STATUS = NONE or CHOICE_STATUS = NOCHOICE;
  case CHOICE is
    — сдвинуть многоугольник в данную позицию
    when SHIFT = >
      REQUEST_LOCATOR (DISPLAY, 1; LOCATOR_STATUS,
        TRANSFORMATION_2, POINT_1);
  exit when LOCATOR_STATUS = NONE;
      REQUEST_LOCATOR (DISPLAY, 1, LOCATOR_STATUS,
        TRANSFORMATION, POINT_2);
  exit when LOCATOR_STATUS = NONE or
        TRANSFORMATION /= TRANSFORMATION_2);
  SELECT_NORMALIZATION_TRANSFORMATION (TRANSFORMATION_2);
  ACCUMULATE_TRANSFORMATION_MATRIX(
    SOURCE_TRANSFORMATION ==> MATRIX,
    FIXED_POINT ==> WC.POINT'((0.0, 0.0)),
    SHIFT_VECTOR ==> WC.VECTOR'
      (POINT_1.X-POINT_2.X,
        POINT_1.Y-POINT_2.Y)),
    ROTATION_ANGLE ==> 0.0,
    SCALE_FACTORS ==> (1.0, 1.0),
    RESULT_TRANSFORMATION ==> MATRIX_RESULT);
  SET_SEGMENT_TRANSFORMATION
  (POLYGON_SEGMENT, MATRIX_RESULT);
  when ZOOM ==> null;
  when ROTATE ==> null;
  when others ==> exit;
  end case;
  UPDATE_WS(DISPLAY, PERFORM);
end loop;
— теперь полигон прорисован
DEACTIVATE_WS (DISPLAY);
DEACTIVATE_WS (SEGSTORE);
OPEN_WS (PLOTTER, PLOT_CONNECTION, PLOT_TYPE);
ACTIVATE_WS (PLOTTER);
— установить представления для этой станции
SET_COLOUR_REPRESENTATION (PLOTTER, RED, (1.0, 0.0, 0.0));

```

```

SET_POLYLINE_REPRESENTATION (PLOTTER, 3, 1, 1.5, RED);
SET_TEXT_REPRESENTATION (PLOTTER, 2, (0, STRING_PRECISION),
                           1.0, 0.0, RED);
SET_WS_VIEWPORT (PLOTTER, (0.0, 0.0, 0.5));
COPY_SEGMENT_TO_WS (PLOTTER, POLYGON_SEGMENT);
SET_TEXT_INDEX (2);
SET_CHAR_HEIGHT (AXIS_CHARACTER_HEIGHT);
TEXT ((0.5, 0.5), «This is polygon»);
  DEACTIVATE_WS (PLOTTER);
  CLOSE_WS (PLOTTER);
  CLOSE_WS (DISPLAY);
  CLOSE_WS (SEGSTORE);
  CLOSE_GKS;
end POLYGON;

```

### B.5. Пример программы 5. PROGRAM SHWLN

— ПРОГРАММА SHOWLN

— ОПИСАНИЕ:

— Данная программа иллюстрирует существующие типы линии на выбранной пользователем станции. Она содержит типичную подпрограмму инициализации ЯГС и демонстрирует подпрограммы, которые не меняют каких-либо элементов списка состояний.

— СОГЛАШЕНИЯ:

— Уровень ЯГС 0a

with GKS\_TYPES;

with GKS;

with TEXT\_IO;

use GKS\_TYPES;

use GKS;

use TEXT\_IO;

procedure SHOWLN is

```

TYPE_OF_WS           : WS_TYPE;
ERROR_IND            : ERROR_NUMBER := 0;
WORKSTATION         : WS_ID := 1;
OP_STATE            : OPERATING_STATE;

```

package WS\_TYPE\_IO is new INTEGER\_IO (WS\_TYPE);

```

procedure INIT_GKS (WTYPE : in out WS_TYPE;
                   ERRIND : in out ERROR_NUMBER) is

```

— Последовательность инициализации ЯГС.

```

ERROR_FILE : constant STRING := 'SHOWLN_ERR_FILE';
GKS_WS_TYPES : WS_TYPES_LIST_OF;
CATEGORY    : WS_CATEGORY;
CONNECTION  : STRING (1..20);
CONN_LENGTH : NATURAL;

```

begin

```
OPEN_GKS (ERROR_FILE);
```

— Запросить доступные типы станций и распечатать их.

```

INQ_LIST_OF_AVAILABLE_WS_TYPES (ERRIND, GKS_WS_TYPES);
if ERRIND /= 0 then

```

```

return;
end if;
PUT_LINE («The available output and outin workstation types are:»);
for I in 1.. WS_TYPES.SIZE_OF_LIST (GKS_WS_TYPES) loop
  INQ_WS_CATEGORY
  (WS_TYPES.LIST_ELEMENT (I, GKS_WS_TYPES), ERRIND, CATEGORY);
  if (CATEGORY = OUTPUT or CATEGORY = OUTIN) then
    WS_TYPE_IO.PUT (WS_TYPES.LIST_ELEMENT (I, GKS_WS_TYPES));
    PUT ("");
  end if;
end loop;
NEW_LINE;

```

— Выберите одну станцию, чтобы открыть и активировать ее.

```

PUT_LINE («Please enter connection identifier and workstation type»);
GET_LINE (CONNECTION, CONN_LENGTH);
WS_TYPE_IO.GET (WTYPE);
OPEN_WS (WORKSTATION, CONNECTION (1.. CONN_LENGTH),
                                                WTYPE);
ACTIVATE_WS (WORKSTATION);

```

— Проверьте режим работы для гарантирования успешных открытий и активирования.

```

INQ_OPERATING_STATE_VALUE (OP_STATE);
if OP_STATE /= WSAC then
  ERRIND := 3;
  return;
end if;
ERRIND := 0;
end INIT_GKS;

```

```

procedure LINE_DEMO (WTYPE      : in out WS_TYPE;
                    ERRIND      : in out ERROR_NUMER) is
  STATUS
  REQ_WINDOW           : UPDATE_STATE;
  CUR_WINDOW           : NDC.RECTANGLE_LIMITS;
  REQ_VIEWPORT         : DC.RECTANGLE_LIMITS;
  CUR_VIEWPORT         : DC.RECTANGLE_LIMITS;
  LINETYPE_LIST        : LINETYPES.LIST_OF;
  NUM_WIDTHS           : NATURAL;
  NOMINAL_WIDTH        : DC.MAGNITUDE;
  RANGE_OF_WIDTHS      : DC.RANGE_OF_MAGNITUDES;
  NUM_INDICES          : NATURAL;
  LIST_OF_ASF          : ASF_LIST := (others
                                     => INDIVIDUAL);
  SAVED_XFORM_NUM      : TRANSFORMATION_NUMBER;
  SAVED_PRIM_ATTR      : PRIMITIVE_ATTRIBUTE_VALUES;
  SAVED_INDV_ATTR      : INDIVIDUAL_ATTRIBUTE_VALUES;
  DISTANCE             : NDC_TYPE;
  PTS                  : WC.POINT_ARRAY (1.. 2);
begin

```

— Проверить режим работы.

```

INQ_OPERATING_STATE_VALUE (OP_STATE);
if (OP_STATE /= WSAC and OP_STATE /= SGOP) then

```

```

ERRIND := 5;
return;
end if;

```

— Узнать преобразование станции.

```

INQ_WS_TRANSFORMATION (WORKSTATION, ERRIND, STATUS,
REQ_WINDOW, CUR_WINDOW, REQ_VIEWPORT, CUR_VIEWPORT);
if ERRIND /= 0 then
return;
end if;

```

— Узнать возможности ломаной.

```

INQ_POLYLINE_FACILITIES (WS_TYPE, ERRIND, LINETYPE_LIST,
NUM_WIDTHS, NOMINAL_WIDTH, NOMINAL_WIDTH,
RANGE_OF_WIDTHS, NUM_INDICES);
if ERRIND /= 0 then
return;
end if;

```

```

INQ_CURRENT_NORMALIZATION_TRANSFORMATION_NUMBER
(ERRIND, SAVED_PRIM_ATTR);
INQ_CURRENT_INDIVIDUAL_ATTRIBUTE_VALUES
(ERRIND, SAVE_INDV_ATTR);

```

— Установить номер преобразования нормирования, флаги выборки индивидуальных атрибутов, масштаб толщины линии (1.0), индекс цвета ломаной (1) и атрибуты приемлемого текста.

```

SELECT_NORMALIZATION_TRANSFORMATION (0);
SET_ASF (LIST_OF_ASF);
SET_LINEWIDTH_SCALE_FACTOR (1.0);
SET_POLYLINE_COLOUR_INDEX (1);
SET_CHAR_UP_VECTOR ((0.0, 1.0));
SET_TEXT_PATH (RIGHT);
SET_TEXT_ALIGNMENT ((LEFT, HALF));
SET_TEXT_FONT_AND_PRECISION ((1, STRING_PRECISION));
SET_CHAR_EXPANSION_FACTOR (1.0);
SET_CHAR_SPACING (0.0);
SET_TEXT_COLOUR_INDEX (1);

```

— Вычислить расстояние между линиями.

```

DISTANCE := (CUR_WINDOW.YMAX - CUR_WINDOW.YMIN) / NDC_TYPE
(LINETYPES.SIZE_OF_LIST (LINETYPE_LIST));

```

— Установить высоту литер, равную половине расстояния между линиями, но не более,  $1/20$  высоты текущей станции.

```

if (DISTANCE/2.0) < ((CUR_WINDOW.YMAX - CUR_WINDOW.YMIN)/20.0)
then SET_CHAR_HEIGHT (WC.MAGNITUDE (DISTANCE/2.0));

```

```

else
SET_CHAR_HEIGHT
(WC.MAGNITUDE ((CUR_WINDOW.YMAX - CUR_WINDOW.YMIN)
/20.0));

```

end if;

— Отступление линии от левой границы к середине окна текущей станции.

```

PTS (1).X := WC_TYPE (CUR_WINDOW.XMIN);

```

```
PTS (1).Y := WC_TYPE (CUR_WINDOW.YMAX - DISTANCE/2.0);
PTS (2).X := WC_TYPE (CUR_WINDOW.XMIN + CUR_WINDOW.XMAX
/2.0);
```

— Цикл по существующим типам линий.

```
for I in 1..LINETYPES.SIZE_OF_LIST (LINETYPE_LIST) loop
  SET_LINETYPE (LINETYPES.LIST_ELEMENT (I, LINETYPE_LIST));
  PTS (2).Y := PTS (1).Y;
  POLYLINE (PTS);
  PTS (1).Y := PTS (1).Y - WC_TYPE (DISTANCE);
```

— Аннотировать тип линии.

```
TEXT (PTS(2,
  INTEGER IMAGE (INTEGER (LINETYPES.LIST_ELEMENT
(I, LINETYPE_LIST) ) ) );
end loop;
```

— Восстановить номер преобразования нормирования и атрибуты.

```
SELECT_NORMALIZATION_TRANSFORMATION (0);
SET_ASF (SAVED_INDV_ATTR.ASF);
SET_LINETHICKNESS_SCALE_FACTOR (SAVED_INDV_ATTR.WIDTH);
SET_POLYLINE_COLOUR_INDEX (SAVED_INDV_ATTR.LINE_COLOUR);
SET_CHAR_UP_VECTOR (SAVED_PRIM_ATTR.CHAR_UP_VECTOR);
SET_TEXT_PATH (SAVED_PRIM_ATTR.PATH);
SET_TEXT_ALIGNMENT (SAVED_PRIM_ATTR.ALIGNMENT);
SET_TEXT_FONT_AND_PRECISION (SAVED_INDV_ATTR.FONT_PRECISION);
```

```
SET_CHAR_EXPANSION_FACTOR (SAVED_INDV_ATTR.EXPANSION);
SET_CHAR_SPACING (SAVED_INDV_ATTR.SPACING);
SET_TEXT_COLOUR_INDEX (SAVED_INDV_ATTR.TEXT_COLOUR);
ERRIND := 0;
end LINE_DEMO;
```

— Главная процедура SHOWLN.

begin

— Вызов процедуры инициализации.

```
INIT_GKS (TYPE_OF_WS, ERROR_IND)
```

```
if ERROR_IND = 0 then
```

— Вызов подпрограммы демонстрации типов линий.

```
LINE_DEMO (TYPE_OF_WS, ERROR_IND);
```

```
end if;
```

— Все закрыть.

```
EMERGENCY_CLOSE_GKS;
```

```
end SHOWLN;
```

## МНОГОЗАДАЧНЫЙ РЕЖИМ РАБОТЫ ЯГС

(Это приложение не является составной частью стандарта, но дает дополнительную информацию).

Встраивание функций ЯГС как подпрограмм в пакет Ада реализует естественным образом: данные «состояния» ЯГС декларируются как переменные, локальные для тела пакета; они непосредственно доступны и модифицируются из подпрограмм ЯГС. Такой подход принимают, когда прикладные программы используют только последовательно управляемые структуры. Проблема состоит в том, что одновременные вызовы программ ЯГС могут испортить переменные состояния; например, при одновременной попытке записать в них. Данная проблема существует, где имеется истинная параллельность (множество процессов) или она моделируется (использование одного процессора с мультиплексированием).

Далее приводится метод реализации, который позволяет преодолеть данную трудность без изменения интерфейса с ЯГС со стороны прикладных программ на языке Ада. Коротко идея состоит в том, чтобы защитить данные тела пакета (например, переменные состояния), локализуя их в задаче, содержащейся в теле пакета. Для каждой подпрограммы, которая обращается к данным, будут существовать соответствующие входы, декларированные в задаче. Одно и то же имя может быть использовано для входа и подпрограммы, применяющей средства совмещения в языке Ада. Тело задачи похоже на монитор, то есть, это цикл, содержащий селективное ожидание с переходом для каждого входа. Предложение приема выполняет действительное считывание или запись информации по состоянию, как требуется соответствующими подпрограммами ЯГС. Тело каждой подпрограммы ЯГС сжимается просто до вызова входа задачи. Таким образом, даже если две задачи из пользовательского прикладного программного обеспечения одновременно вызывают подпрограммы, которые модифицируют или обращаются к переменным состояниям, это будет приводить к вызовам входа задачи, которые ставятся в очередь и обслуживаются в порядке поступления. В данном случае опасность порчи переменных состояний отсутствует.

Для иллюстрации данного метода в следующем примере показан, как может быть написан скелет пакета ЯГС.

```
with GKS_TYPES;
use GKS_TYPES;
package GKS is;

  procedure OPEN_GKS
    (ERROR_FILE           : in STRING :=DEFAULT_ERROR_
                                     _FILE;
    AMOUNT_OF_MEMORY     : in NATURAL :=DEFAULT_MEMORY
                                     _UNITS);

  procedure OPEN_WS
    (WS_CONNECTION       : in WS_ID;
    WS_TYPE_OF_WS        : in STRING;
    WS_ID_OF_WS           : in WS_TYPE);

  procedure CLOSE_GKS;
```

end GKS;

— Версия для последовательных прикладных программ:

```
with ERROR_HANDLING;
package body GKS is
```

— Переменные состояния:

```
CURRENT_OPERATING_STATE : OPERATING_STATE := GKCL;
SET_OF_OPEN_WORKSTATIONS : WS_IDS.LIST_OF := WS_IDS.NULL_LIST;
```

```
procedure OPEN_GKS
(ERROR_FILE           : in STRING := DEFAULT_ERROR_
                        _FILE;
AMOUNT_OF_MEMORY     : in NATURAL := DEFAULT_
                        _MEMORY_UNITS) is
```

```
begin
```

```
if CURRENT_OPERATING_STATE /= GKCL then
    ERROR_HANDLING (1, «OPEN_GKS»);
```

```
else
    CURRENT_OPERATING_STATE := GKOP;
end if;
```

```
end OPEN_GKS;
procedure OPEN_WS
```

```
(WS           : in WS_ID;
CONNECTION    : in STRING;
TYPE_OF_WS    : in WS_TYPE) is
```

```
begin
```

```
if CURRENT_OPERATING_STATE not in GKOP .. SGOP then
    ERROR_HANDLING (8, «OPEN_WS»);
```

```
else
    CURRENT_OPERATING_STATE := WSOP;
    WS_IDS.ADD_TO_LIST(WS, SET_OF_OPEN_WORKSTATIONS);
end if;
```

```
end OPEN_WS;
procedure CLOSE_GKS is
begin
```

```
if CURRENT_OPERATING_STATE /= GKOP then
    ERROR_HANDLING (2, «CLOSE_GKS»);
```

```
else
    CURRENT_OPERATING_STATE := GKCL;
end if;
```

```
end CLOSE_GKS;
```

```
end GKS;
```

— Версия для прикладных программ, использующих многозадачный режим работы:

```

with ERROR_HANDLING;
package body GKS is
task MONITOR is
entry OPEN_GKS
  (ERROR_FILE           : in STRING :=DEFAULT_ERROR_
                           _FILE;
   AMOUNT_OF_MEMORY     : in NATURAL :=DEFAULT_MEMORY
                           _UNITS);

entry OPEN_WS
  (WS                    : in WS_ID;
   CONNECTION           : in STRING;
   TYPE_OF_WS          : in WS_TYPE);

entry CLOSE_GKS;
...
end MONITOR;
task body MONITOR is

— Переменные состояния:
CURRENT_OPERATING_STATE :=OPERATING_STATE =GKCL;
SET_OF_OPEN_WORKSTATIONS :=WS_IDS.LIST_OF
                           :=WS_IDS.NULL_LIST;

begin
loop
begin
select
accept OPEN_GKS
  (ERROR_FILE           : in STRING :=DEFAULT_ERROR_
                           _FILE;
   AMOUNT_OF_MEMORY     : in NATURAL :=DEFAULT_MEMORY
                           _UNITS);

do
if CURRENT_OPERATING_STATE/=GKCL then
  ERROR_HANDLING (1, «OPEN_GKS»);
else
  CURRENT_OPERATING_STATE := GKOP;
end if;

end OPEN_GKS;
or
accept OPEN_WS
  (WS                    : in WS_ID;
   CONNECTION           : in STRING;
   TYPE_OF_WS          : in WS_TYPE) do
...
if CURRENT_OPERATING_STATE /= not in GKOP..SGOP then
  ERROR_HANDLING (8, «OPEN_WS»);
else
  CURRENT_OPERATING_STATE := WSOP;
  WS_IDS.ADD_TO_LIST (WS, SET_OF_OPEN_WORKSTATIONS);
end if;
...

```

```

end OPEN_WS;
or
accept CLOSE_GKS do
  if CURRENT_OPERATING_STATE /= GKOP then
    ERROR_HANDLING (2, «CLOSE_GKS»);
  else
    CURRENT_OPERATING_STATE := GKCL;
  end if;
end CLOSE_GKS;
or
terminate;
end select;
exception
  when others => null;
end;
end loop;
end MONITOR;
procedure OPEN_GKS
  (ERROR_FILE           : in STRING := DEFAULT_ERROR_
                           _FILE;
   AMOUNT_OF_MEMORY    : in NATURAL := DEFAULT_MEMORY
                           _UNITS) is
begin
  MONITOR.OPEN_GKS(ERROR_FILE, AMOUNT_OF_MEMORY);
end OPEN_GKS;
  (WS                     : in WS_ID;
   CONNECTION            : in STRING;
   TYPE_OF_WS           : in WS_TYPE) is
begin
  MONITOR.OPEN_WS(WS, CONNECTION, TYPE_OF_WS);
end OPEN_WS;
procedure CLOSE_GKS is
begin
  MONITOR.CLOSE_GKS;
end CLOSE_GKS;
end GKS;

```

Несколько замечаний по взаимодействию с обработкой прерываний событий в случае, когда процедура `ERROR_HANDLING` вызывает `GKS_ERROR`. Отметим, что в обеих версиях тела пакета вызывается процедура `ERROR_HANDLING`. Предположим, что прикладная программа вызывает `OPEN_GKS`, когда ЯГС уже открыт. В последовательной версии вызов `ERROR_HANDLING` из тела процедуры `OPEN_GKS` вызовет распространение `GKS_ERROR` назад к прикладной программе, которая вызвала `OPEN_GKS`. В версии для многозадачного режима работы тот же самый эффект будет достигнут следующим образом. В течение выполнения предложения приема для входа `OPEN_GKS` будет вызвана процедура `ERROR_HANDLING` и произойдет `GKS_ERROR`. В соответствии с семантикой языка Ада, так как это исключение не обрабатывается локальным обработчиком при приеме, оно распространяется (1) в точку, следующую за приемом, и (2) в точку вызова входа. В первом случае оно обрабатывается в

блоке, включающем в себя предложение выбора; таким образом задача монитор может перейти к следующей итерации без разрыва. Для второго случая точка находится в теле процедуры OPEN\_GKS. Так как здесь нет обработчика исключительных событий, то GKS\_ERROR распространяется, как это и хотелось, назад к точке вызова в прикладной программе.

Завершение выполнения задачи MONITOR будет реализовано выбором альтернативы завершения, когда задачи прикладной программы закончатся.

Существует ряд вариаций метода, рассмотренного и проиллюстрированного выше, которые при реализации могут представлять интерес с точки зрения повышения потенциального параллелизма прикладных программ ЯГС, использующих многозадачный режим работы. С помощью только что описанного метода единый набор переменных состояния защищается одной задачей. Если информация о состоянии может быть разделена на независимые наборы с одной задачей монитором на набор, то задача прикладной программы, считывающая/записывающая переменные в один набор, может выполнять это одновременно с задачей, которая считывает/записывает в другой набор.

Другой вариант защиты переменных состояния состоит в том, чтобы различить функции ЯГС, просто считывающие значения данных, от тех, которые записывают их. Метод, описанный выше, рассматривает «читателей» и «писателей» одинаково; таким образом он запрещает двум задачам прикладной программы одновременно считывать информацию о состоянии. Можно написать программу монитор таким образом, чтобы допускалось одновременное считывание задачами прикладной программы и запрещалось одновременное записывание и одновременное считывание и записывание. Существует множество различных подходов к решению данной задачи, зависящих от учета программистом таких факторов, как:

- возможность задачи прикладной программы быть преждевременно прерванной;

- возможность задачи прикладной программы «голодать» по обслуживанию из-за того, как запрограммирован прием входов.

Реализация ЯГС, которые поддерживают многозадачные программы на языке Ада, могут быть использованы для программ, которые являются полностью последовательными, хотя эффективность при этом может несколько снизиться. При реализации может также возникнуть идея предоставить два тела пакета ЯГС: одно для последовательных применений, другое — для многозадачных.

## НЕПОДДЕРЖИВАЕМЫЕ ОБОБЩЕННЫЕ ПРИМИТИВЫ ВЫВОДА И РАСШИРЕНИЯ

(Это приложение не является составной частью стандарта,  
но дает дополнительную информацию.)

В настоящем приложении проясняются взаимоотношения между метафайлом GKSM и обобщенными примитивами вывода (ОПВ) и расширениями (ESC). Каждая функция ОПВ и ESC, являющаяся зарегистрированной, доступна для прикладных программ как отдельная процедура со своими собственными формальными параметрами и именем подпрограммы, как описано в п. 5.1.

Реализация ЯГС/Ада должна предоставлять возможность записывать и считывать зарегистрированные ОПВ и ESC в метафайл, даже если реализация не поддерживает функций ОПВ и ESC. Следовательно, чтобы данная возможность поддерживалась, форматы записей данных для зарегистрированных ОПВ и ESC должны существовать между реализациями.

Например, пусть метафайл А генерируется в реализации ЯГС/Ада, которая поддерживает ОПВ-окружность. Метафайл А теперь имеет запись данных, содержащую идентификатор окружности, точку центра и радиус. Далее метафайл А переносится в другую реализацию, которая не поддерживает ОПВ-окружность. В новой среде будет генерироваться метафайл В, содержащий весь метафайл А и дополнительные графические данные. И важно, чтобы ОПВ-окружность из метафайла А была включен в метафайл В, даже если на одной станции в этой среде не способна генерировать изображение окружности.

Для иллюстрации метода в следующем примере показаны фрагмент прикладного кода и реализация функции метафайла INTERPRET-ITEM.

```
with GKS;
use GKS;
with GKS-TYPES;
use GKS-TYPES;
```

— Данная прикладная программа передает данные из метафайла А в метафайл В.

```
procedure TRANSFER_METAFILE is
```

— описание переменных

```
INPUT_METAFILE           : constant WS_ID := 1;
OUTPUT_METAFILE          : constant WS_ID := 2;
INPUT_METAFILE_TYPE      : constant WS_TYPE := 2;
OUTPUT_METAFILE_TYPE     : constant WS_TYPE := 3;
INPUT_METAFILE_CONNECTION_ID
                          : constant STRING := «METAFILE_A»;
OUTPUT_METAFILE_CONNECTION_ID
                          : constant STRING := «METAFILE_B»;
METAFILE_DATA_RECORD     : GKSM_DATA_RECORD;
METAFILE_ITEM_TYPE       : GKSM_ITEM_TYPE;
LENGTH_MAX_LENGTH       : NATURAL := 500;
ERROR_FILE : constant STRING := «MY_ERROR_FILE»;
```

```
begin
```

```

— Открыть ЯГС.
  OPEN_GKS (ERROR_FILE);

— Открыть метафайлы ввода и вывода.
  OPEN_WS (INPUT_METAFILE,
           INPUT_METAFILE_CONNECTION_ID,
           INPUT_METAFILE_TYPE);

  OPEN_WS (OUTPUT_METAFILE,
           OUTPUT_METAFILE_CONNECTION_ID,
           OUTPUT_METAFILE_TYPE);

— Активизировать только метафайл ввода.
  ACTIVATE_WS (OUTPUT_METAFILE);

— В данном цикле каждый элемент метафайла А считывается и передается в
ЯГС посредством вызова функции INTERPRET_ITEM. Помните, что метафайл
А содержит ОПВ-окружности, обрабатываемые функцией INTERPRET_ITEM,
которая представлена далее в примере.
loop
  GET_ITEM_TYPE_FROM_GKSM (INPUT_METAFILE,
                           METAFILE_ITEM_TYPE, LENGTH);
  if METAFILE_ITEM_TYPE=0 then
    exit;      — выйти из цикла, метафайл полон.
  end if;
  READ_ITEM_FROM_GKSM (INPUT_METAFILE, MAX_LENGTH,
                       METAFILE_DATA_RECORD);
  INTERPRET_ITEM (METAFILE_DATA_RECORD);
end loop;

— Деактивируется только выходной файл.
DEACTIVATE_WS (OUTPUT_METAFILE);

— Закрыть метафайлы ввода и вывода.
CLOSE_WS (INPUT_METAFILE);
CLOSE_WS (OUTPUT_METAFILE);

— Закрыть GKS.
CLOSE_GKS;
end TRANSFER_METAFILE;

— Пример прикладной программы опирается на реализацию функции
INTERPRET_ITEM для распознавания ОПВ-окружности и передачи ОПВ в
выходной метафайл. Это легко может быть выполнено следующим сегментом
кода.
— Предположим, что личный тип GKSM_DATA_RECORD декларирован как
тип записи дискриминанты с различными компонентами, основанными на типе
элементов. Когда GKSM_DATA_RECORD содержит ОПВ, имеются различные
поля, содержащие всю существующую информацию о ОПВ.
type GKSM_DATA_RECORD      : GKSM_ITEM_TYPE := 0;
      (TYPE_OF_ITEM       : NATURAL := 0) is
      LENGTH
record
  case TYPE_OF_ITEM is
    when OPEN_GKS      => ...
    when POLYLINE     => ...
    when GDP           => ...

```

```

ID : GDP_ID;
NUM_PTS : POSITIVE;
INTEGER_DATA_LENGTH : NATURAL;
REAL_DATA_LENGTH : NATURAL;
LIST_OF_POINTS : WC.POINT_ARRAY (1.. NUM_PTS);
INTEGER_DATA : INTEGER_ARRAY
(1.. INTEGER_DATA_LENGTH);
REAL_DATA : REAL_ARRAY (1.. REAL_DATA_
                        _LENGTH);

```

```

when ...
end case;
end record;

```

— Пример того, как реализация могла бы поддержать передачу нереализованного ОПВ через функцию INTERPRET\_ITEM.

```

procedure INTERPRET_ITEM (ITEM : in GKSM_DATA_RECORD) is
REGISTERED_GDP_CIRCLE : constant GDP_ID := 1;

```

```

begin
case ITEM TYPE_OF_ITEM is
when OPEN_GKS => ...
when POLYLINE => ...
when GDP =>
. case ITEM ID is
when REGISTERED_GDP_SPLINE => ...
when REGISTERED_GDP_ELLIPSE => ...
when REGISTERED_GDP_CIRCLE =>

```

— Вызвать генератор метафайла, в котором запись данных ITEM как параметр записывается во все открытые и активные метафайлы.

```

when ...
end case;
end INTERPRET_ITEM;

```

## ТИПЫ ЭЛЕМЕНТОВ МЕТАФАЙЛА

(Это приложение не является составной частью стандарта, но дает дополнительную информацию.)

Функция GET ITEM TYPE FROM GKSM возвращает тип следующего элемента метафайла, однако значение данного типа может изменяться в зависимости от реализации метафайла. Для того, чтобы можно было писать программы, независимые от реализации метафайла, предлагаются имена в Аде (см. таблицу). В реализации следует задать эти имена со значениями, которые соответствуют значениям, возвращаемым процедурой GET ITEM TYPE FROM GKSM.

Тип элемента GKSM	Имя в Аде
ФЛАГИ ВЫБОРКИ АТРИБУТОВ МАТРИЦА ЯЧЕЕК МАСШТАБ РАСШИРЕНИЯ ЛИТЕР	GKSM_ASF GKSM_CELL_ARRAY GKSM_CHAR_EXPANSION_
МЕЖЛИТЕРНЫЙ ПРОСВЕТ ВЕКТОР ЛИТЕРЫ ОЧИСТИТЬ СТАНЦИЮ ПРЯМОУГОЛЬНИК ОТСЕЧЕНИЯ ЗАКРЫТЬ СЕГМЕНТ ПРЕДСТАВЛЕНИЕ ЦВЕТА	_FACTOR GKSM_CHAR_SPACING GKSM_CHAR_VECTORS GKSM_CLEAR_WS GKSM_CLIPPING_RECTANGLE GKSM_CLOSE_SEGMENT GKSM_COLOUR_
СОЗДАТЬ СЕГМЕНТ РЕЖИМ ЗАДЕРЖКИ УНИЧТОЖИТЬ СЕГМЕНТ КОНЕЦ ЗАПИСИ РАСШИРЕНИЕ ПОЛИГОНАЛЬНАЯ ОБЛАСТЬ ИНДЕКС ЦВЕТА ПОЛИГОНАЛЬНОЙ ОБЛАСТИ ИНДЕКС ПОЛИГОНАЛЬНОЙ ОБЛАСТИ	_REPRESENTATION GKSM_CREATE_SEGMENT GKSM_DEFERRAL_STATE GKSM_DELETE_SEGMENT GKSM_END_ITEM GKSM_ESCAPE GKSM_FILL_AREA GKSM_FILL_AREA_COLOUR_
ВИД ЗАПОЛНЕНИЯ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ ПРЕДСТАВЛЕНИЕ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ ИНДЕКС ЗАПОЛНИТЕЛЯ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ ОБОБЩЕННЫЙ ПРИМИТИВ ВЫВОДА (ОПВ)	_INDEX GKSM_FILL_AREA_INDEX GKSM_FILL_AREA_INTERior_
ТИП ЛИНИИ МАСШТАБ ШИРИНЫ ЛИНИИ	_STYLE GKSM_FILL_AREA_           _REPRESENTATION GKSM_FILL_AREA_STYLE_           _INDEX GKSM_GDP GKSM_LINETYPE GKSM_LINewidth_SCALE_           _FACTOR

Тип элемента GKSM	Имя в Аде
МАСШТАБ РАЗМЕРА МАРКЕРА	GKSM_MARKER_SIZE_SCALE_
ТИП МАРКЕРА	_FACTOR
СООБЩЕНИЕ	GKSM_MARKER_TYPE
ОПОРНАЯ ТОЧКА ШАБЛОНА	GKSM_MESSAGE
ПРЕДСТАВЛЕНИЕ ШАБЛОНА	GKSM_PATTERN_REFERENCE_
ВЕКТОРА ШАБЛОНА	_POINT
ИДЕНТИФИКАТОР ВЫБОРА	GKSM_PATTERN_
ЛОМАНАЯ	_REPRESENTATION
ИНДЕКС ЦВЕТА ЛОМАНОЙ	GKSM_PATTERN_VECTORS
ИНДЕКС ЛОМАНОЙ	GKSM_PICK_ID
ПРЕДСТАВЛЕНИЕ ЛОМАНОЙ	GKSM_POLYLINE
ПОЛИМАРКЕР	GKSM_POLYLINE_COLOUR_
ИНДЕКС ЦВЕТА ПОЛИМАРКЕРА	_INDEX
ПРЕДСТАВЛЕНИЕ ПОЛИМАРКЕРА	GKSM_POLYLINE_INDEX
ПЕРЕРИСОВАТЬ ВСЕ СЕГМЕНТЫ	GKSM_POLYLINE_
НА СТАНЦИИ	_REPRESENTATION
ПЕРЕИМЕНОВАТЬ СЕГМЕНТЫ	GKSM_POLYMARKER
ЗАДАТЬ ЧУВСТВИТЕЛЬНОСТЬ К	GKSM_POLYMARKER_COLOUR_
УКАЗАНИЮ	_INDEX
ЗАДАТЬ ВЫДЕЛЕНИЕ	GKSM_POLYMARKER_
ЗАДАТЬ ПРИОРИТЕТ СЕГМЕНТА	_REPRESENTATION
ЗАДАТЬ ПРЕОБРАЗОВАНИЕ СЕГ-	GKSM_REDRAW_ALL_
МЕНТА	_SEGMENT_WS
ЗАДАТЬ ВИДИМОСТЬ	GKSM_RENAME_SEGMENT
ТЕКСТ	GKSM_SET_DETECTABILITY
ВЫРАВНИВАНИЕ ТЕКСТА	GKSM_SET_HIGHLIGHTING
ИНДЕКС ЦВЕТА ТЕКСТА	GKSM_SET_SEGMENT_PRIORITY
ШРИФТ И ТОЧНОСТЬ ТЕКСТА	GKSM_SET_SEGMENT_
ИНДЕКС ТЕКСТА	_TRANSFORMATION
НАПРАВЛЕНИЕ ТЕКСТА	GKSM_SET_VISIBILITY
ПРЕДСТАВЛЕНИЕ ТЕКСТА	GKSM_TEXT
ОБНОВИТЬ ИЗОБРАЖЕНИЕ НА	GKSM_TEXT_ALIGNMENT
СТАНЦИИ	GKSM_TEXT_COLOUR_INDEX
НАЧАЛО ЗАПИСИ ПОЛЬЗОВАТЕЛЯ	GKSM_TEXT_FONT_AND_
ПОЛЕ ВЫВОДА СТАНЦИИ	_PRECISION
ОКНО СТАНЦИИ	GKSM_TEXT_INDEX
	GKSM_TEXT_PATH
	GKSM_TEXT_REPRESENTATION
	GKSM_UPDATE_WS
	GKSM_USER_ITEM
	GKSM_WS_VIEWPORT
	GKSM_WS_WINDOW

## ИНДЕКСЫ ФУНКЦИЙ ЯГС

(Это приложение не является составной частью стандарта,  
но дает дополнительную информацию.)

## Ж.1. Функции ЯГС

ACCUMULATE TRANSFORMATION MATRIX	93
ACTIVATE WORKSTATION	52
ASSOCIATE SEGMENT WITH WORKSTATION	62
AWAIT EVENT	69
CELL ARRAY	55
CLEAR WORKSTATION	53
CLOSE GKS	52
CLOSE SEGMENT	62
CLOSE WORKSTATION	52
COPY SEGMENT TO WORKSTATION	63
CREATE SEGMENT	62
DEACTIVATE WORKSTATION	53
DELETE SEGMENT	62
DELETE SEGMENT FROM WORKSTATION	62
EMERGENCY CLOSE GKS	94
ERROR HANDLING	4, 5, 94
ERROR LOGGING	5, 94
ESCAPE	53
EVALUATE TRANSFORMATION MATRIX	93
FILL AREA	55
FLUSH DEVICE EVENTS	69
GENERALIZED DRAWING PRIMITIVE	55
GET CHOICE	70
GET ITEM TYPE FROM GKSM	71
GET LOCATOR	70
GET PICK	76
GET STRING	79
GET STROKE	70
GET VALUATOR	70
INITIALISE CHOICE	65
INITIALISE LOCATOR	64
INITIALISE PICK	65
INITIALISE STRING	65
INITIALISE STROKE	64
INITIALISE VALUATOR	64
INQUIRE CHOICE DEVICE STATE	81
INQUIRE CLIPPING	81
INQUIRE COLOUR FACILITIES	87
INQUIRE COLOUR REPRESENTATION	79
INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES	4, 73
INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER	75
INQUIRE CURRENT PICK_IDENTIFIER VALUE	73

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES	4, 72
INQUIRE DEFAULT CHOICE DEVICE DATA	90
INQUIRE DEFAULT DEFERRAL STATE VALUES	84
INQUIRE DEFAULT LOCATOR DEVICE DATA	89
INQUIRE DEFAULT PICK DEVICE DATA	91
INQUIRE DEFAULT STRING DEVICE DATA	91
INQUIRE DEFAULT STROKE DEVICE DATA	90
INQUIRE DEFAULT VALUATOR DEVICE DATA	90
INQUIRE DISPLAY SPACE SIZE	83
INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES	88
INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES	83
INQUIRE FILL AREA FACILITIES	86
INQUIRE FILL AREA REPRESENTATION	79
INQUIRE GENERALIZED DRAWING PRIMITIVE	87
INQUIRE INPUT QUEUE OVERFLOW	92
INQUIRE LEVEL OF GKS	71
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES	87
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES	71
INQUIRE LIST OF COLOUR INDICES	79
INQUIRE LIST OF FILL AREA INDICES	78
INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS	75
INQUIRE LIST OF PATTERN INDICES	79
INQUIRE LIST OF POLYLINE INDICES	77
INQUIRE LIST OF POLYMARKER INDICES	77
INQUIRE LIST OF TEXT INDICES	78
INQUIRE LOCATOR DEVICE STATE	80
INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES	88
INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER	72
INQUIRE MORE SIMULTANEOUS EVENTS	76
INQUIRE NAME OF OPEN SEGMENT	75
INQUIRE NORMALIZATION TRANSFORMATION	75
INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES	89
INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED	88
INQUIRE OPERATING STATE VALUE	71
INQUIRE PATTERN FACILITIES	86
INQUIRE PATTERN REPRESENTATION	79
INQUIRE PICK DEVICE STATE	82
INQUIRE PIXEL	92
INQUIRE PIXEL ARRAY DIMENSION	92
INQUIRE PIXEL ARRAY	92
INQUIRE POLYLINE FACILITIES	84
INQUIRE POLYLINE REPRESENTATION	77
INQUIRE POLYMARKER FACILITIES	94
INQUIRE POLYMARKER REPRESENTATION	77
INQUIRE PREDEFINED COLOUR REPRESENTATION	87
INQUIRE PREDEFINED FILL AREA REPRESENTATION	86
INQUIRE PREDEFINED PATTERN REPRESENTATION	86
INQUIRE PREDEFINED POLYLINE REPRESENTATION	84
INQUIRE PREDEFINED POLYMARKER REPRESENTATION	85
INQUIRE PREDEFINED TEXT REPRESENTATION	85
INQUIRE SEGMENT ATTRIBUTES	91
INQUIRE SET OF ACTIVE WORKSTATIONS	72
INQUIRE SET OF ASSOCIATED WORKSTATIONS	91

INQUIRE SET OF OPEN WORKSTATIONS	72
INQUIRE SET OF SEGMENT NAMES IN USE	76
INQUIRE SET OF SEGMENT NAMES ON WORKSTATION	80
INQUIRE STRING DEVICE STATE	82
INQUIRE STROKE DEVICE STATE	81
INQUIRE TEXT EXTENT	78
INQUIRE TEXT FACILITIES	83
INQUIRE TEXT REPRESENTATION	78
INQUIRE VALUATOR DEVICE STATE	81
INQUIRE WORKSTATION CATEGORY	82
INQUIRE WORKSTATION CLASSIFICATION	83
INQUIRE WORKSTATION CONNECTION AND TYPE	76
INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES	76
INQUIRE WORKSTATION MAXIMUM NUMBERS	71
INQUIRE WORKSTATION STATE	78
INQUIRE WORKSTATION TRANSFORMATION	80
INSERT SEGMENT	63
INTERPRET ITEM	71
MESSAGE	53
OPEN GKS	52
OPEN WORKSTATION	52
POLYLINE	55
POLYMARKER	55
READ ITEM FROM GKSM	71
REDRAW ALL SEGMENTS ON WORKSTATION	53
RENAME SEGMENT	62
REQUEST CHOICE	67
REQUEST LOCATOR	66
REQUEST PICK	67
REQUEST STRING	68
REQUEST STROKE	67
REQUEST VALUATOR	67
SAMPLE CHOICE	69
SAMPLE LOCATOR	68
SAMPLE PICK	69
SAMPLE STRING	69
SAMPLE STROKE	68
SAMPLE VALUATOR	68
SELECT NORMALIZATION TRANSFORMATION	61
SET ASPECT SOURCE FLAGS	59
SET CHARACTER EXPANSION FACTOR	58
SET CHARACTER HEIGHT	58
SET CHARACTER SPACING	58
SET CHARACTER UP VECTOR	58
SET CHOICE MODE	60
SET CLIPPING INDICATOR	61
SET COLOUR REPRESENTATION	60
SET DEFERRAL STATE	53
SET DETECTABILITY	63
SET FILL AREA COLOUR INDEX	58

SET FILL AREA INDEX	58
SET FILL AREA INTERIOR STYLE	59
SET FILL AREA REPRESENTATION	60
SET FILL AREA STYLE INDEX	59
SET HIGHLIGHTING	63
SET LINETYPE	57
SET LINewidth SCALE FACTOR	57
SET LOCATOR MODE	65
SET MARKER SIZE SCALE FACTOR	57
SET MARKER TYPE	57
SET PATTERN REFERENCE POINT	59
SET LINewidth SCALE FACTOR	57
SET LOCATOR MODE	65
SET MARKER SIZE SCALE FACTOR	57
SET MARKER TYPE	57
SET PATTERN REFERENCE POINT	59
SET PATTERN REPRESENTATION	60
SET PATTERN SIZE	59
SET PICK IDENTIFIER	59
SET PICK MODE	66
SET POLYLINE COLOUR INDEX	57
SET POLYLINE INDEX	57
SET POLYLINE REPRESENTATION	59
SET POLYMARKER COLOUR INDEX	57
SET POLYMARKER INDEX	57
SET POLYMARKER REPRESENTATION	60
SET SEGMENT PRIORITY	63
SET SEGMENT TRANSFORMATION	63
SET STRING MODE	66
SET STROKE MODE	65
SET TEXT ALIGNMENT	58
SET TEXT FONT AND PRECISION	58
SET TEXT INDEX	57
SET TEXT PATH	58
SET TEXT REPRESENTATION	60
SET VALUATOR MODE	66
SET VIEWPORT	61
SET VIEWPORT INPUT PRIORITY	61
SET VISIBILITY	63
SET WINDOW	61
SET WORKSTATION VIEWPORT	62
SET WORKSTATION WINDOW	61
TEXT	55
UPDATE WORKSTATION	53
WRITE ITEM TO GKSM	70
Ж.2. Процедуры языка Ада	
ACCUMULATE_TRANSFORMATION_MATRIX	93
ACTIVATE_WS	52
ASSOCIATE_SEGMENT_WITH_WS	62
AWAIT_EVENT	69

CELL_ARRAY	55
CLEAR_WS	53
CLOSE_GKS	52
CLOSE_SEGMENT	62
CLOSE_WS	52
COPY_SEGMENT_TO_WS	63
CREATE_SEGMENT	62
DEACTIVATE_WS	53
DELETE_SEGMENT	62
DELETE_SEGMENT_FROM_WS	62
EMERGENCY_CLOSE_GKS	94
ERROR_HANDLING	4, 5, 94
ERROR_LOGGING	5, 94
ESCAPE	53
EVALUATE_TRANSFORMATION_MATRIX	93
FILL_AREA	55
FLUSH_DEVICE_EVENTS	69
GDP	55
GET_CHOICE	70
GET_ITEM_TYPE_FROM_GKSM	71
GET_LOCATOR	70
GET_PICK	70
GET_STRING	70
GET_STROKE	70
GET_VALUATOR	70
INITIALISE_CHOICE	64
INITIALISE_LOCATOR	64
INITIALISE_PICK	65
INITIALISE_STRING	65
INITIALISE_STROKE	64
INITIALISE_VALUATOR	64
INQ_CHAR_BASE_VECTOR	130
INQ_CHAR_EXPANSION_FACTOR	131
INQ_CHAR_HEIGHT	129
INQ_CHAR_SPACING	131
INQ_CHAR_UP_VECTOR	129
INQ_CHAR_WIDTH	129
INQ_CHOICE_DEVICE_STATE	81
INQ_CLIPPING	73
INQ_COLOUR_FACILITIES	79
INQ_COLOUR_REPRESENTATION	4, 73
INQ_CURRENT_INDIVIDUAL_ATTRIBUTE_VALUES	74
INQ_CURRENT_NORMALIZATION_TRANSFORMATION_NUMBER	75
INQ_CURRENT_PICK_ID_VALUE	73
INQ_CURRENT_PRIMITIVE_ATTRIBUTE_VALUES	4, 72
INQ_DEFAULT_CHOICE_DEVICE_DATA	90
INQ_DEFAULT_DEFERRAL_STATE_VALUE	84
INQ_DEFAULT_LOCATOR_DEVICE_DATA	89
INQ_DEFAULT_PICK_DEVICE_DATA	91
INQ_DEFAULT_STRING_DEVICE_DATA	9E

INQ_DEFAULT_STROKE_DEVICE_DATA	90
INQ_DEFAULT_VALUATOR_DEVICE_DATA	90
INQ_DISPLAY_SPACE_SIZE	83
INQ_DYNAMIC_MODIFICATION_OF_SEGMENT_ATTRIBUTES	88
INQ_DYNAMIC_MODIFICATION_OF_WS_ATTRIBUTES	83
INQ_FILL_AREA_COLOUR_INDEX	78
INQ_FILL_AREA_FACILITIES	86
INQ_FILL_AREA_INDEX	78
INQ_FILL_AREA_INTERIOR_STYLE	131
INQ_FILL_AREA_REPRESENTATION	133
INQ_FILL_AREA_STYLE_INDEX	133
INQ_GDP	87
INQ_INPUT_QUEUE_OVERFLOW	140
INQ_LEVEL_OF_GKS	71
INQ_LINETYPE	130
INQ_LINewidth_SCALE_FACTOR	130
INQ_LIST_OF_ASF	131
INQ_LIST_OF_AVAILABLE_GDP	137
INQ_LIST_OF_AVAILABLE_WS_TYPES	129
INQ_LIST_OF_COLOUR_INDICES	133
INQ_LIST_OF_FILL_AREA_INDICES	133
INQ_LIST_OF_NORMALIZATION_TRANSFORMATION_NUMBERS	131
INQ_LIST_OF_PATTERN_INDICES	79
INQ_LIST_OF_POLYLINE_INDICES	77
INQ_LIST_OF_POLYMARKER_INDICES	77
INQ_LIST_OF_TEXT_INDICES	78
INQ_LOCATOR_DEVICE_STATE	80
INQ_MAX_LENGTH_OF_WS_STATE_TABLES	88
INQ_MAX_NORMALIZATION_TRANSFORMATION_NUMBER	72
INQ_MORE_SIMULTANEOUS_EVENTS	76
INQ_NAME_OF_OPEN_SEGMENT	75
INQ_NORMALIZATION_TRANSFORMATION	75
INQ_NUMBER_OF_AVAILABLE_LOGICAL_INPUT_DEVICES	88
INQ_NUMBER_OF_SEGMENT_PRIORITIES_SUPPORTED	88
INQ_OPERATING_STATE_VALUE	71
INQ_PATTERN_FACILITIES	86
INQ_PATTERN_HEIGHT_VECTOR	73
INQ_PATTERN_REFERENCE_POINT	73
INQ_PATTERN_REPRESENTATION	79
INQ_PATTERN_WIDTH_VECTOR	73
INQ_PICK_DEVICE_STATE	73
INQ_PIXEL	92
INQ_PIXEL_ARRAY	92
INQ_PIXEL_ARRAY_DIMENSION	92
INQ_POLYLINE_COLOUR_INDEX	74
INQ_POLYLINE_FACILITIES	84
INQ_POLYLINE_INDEX	72
INQ_POLYLINE_REPRESENTATION	77
INQ_POLYMARKER_COLOUR_INDEX	74
INQ_POLYMARKER_FACILITIES	136
INQ_POLYMARKER_INDEX	129
INQ_POLYMARKER_REPRESENTATION	77
INQ_POLYMARKER_SIZE_SCALE_FACTOR	74
INQ_POLYMARKER_TYPE	74

INQ_PREDEFINED_COLOUR_REPRESENTATION	87
INQ_PREDEFINED_FILL_AREA_REPRESENTATION	86
INQ_PREDEFINED_PATTERN_REPRESENTATION	86
INQ_PREDEFINED_POLYLINE_REPRESENTATION	84
INQ_PREDEFINED_POLYMARKER_REPRESENTATION	85
INQ_PREDEFINED_TEXT_REPRESENTATION	85
INQ_SEGMENT_ATTRIBUTES	91
INQ_SET_OF_ACTIVE_WS	72
INQ_SET_OF_ASSOCIATED_W	91
INQ_SET_OF_OPEN_WS	72
INQ_SET_OF_SEGMENT_NAMES_IN_USE	76
INQ_SET_OF_SEGMENT_NAMES_ON_WS	80
INQ_STRING_DEVICE_STATE	82
INQ_STROKE_DEVICE_STATE	81
INQ_TEXT_ALIGNMENT	73
INQ_TEXT_COLOUR_INDEX	74
INQ_TEXT_EXTENT	78
INQ_TEXT_FACILITIES	85
INQ_TEXT_FONT_AND_PRECISION	74
INQ_TEXT_INDEX	72
INQ_TEXT_PATH	73
INQ_TEXT_REPRESENTATION	78
INQ_VALUATOR_DEVICE_STATE	81
INQ_WS_CATEGORY	82
INQ_WS_CLASSIFICATION	83
INQ_WS_CONNECTION_AND_TYPE	76
INQ_WS_DEFERRAL_AND_UPDATE_STATES	76
INQ_WS_MAX_NUMBERS	76
INQ_WS_STATE	76
INQ_WS_TRANSFORMATION	80
INSERT_SEGMENT	63
INTERPRET_ITEM	71
MESSAGE	53
OPEN_GKS	52
OPEN_WS	52
POLYLINE	55
POLYMARKER	55
READ_ITEM_FROM_GKSM	71
REDRAW_ALL_SEGMENTS_ON_WS	53
RENAME_SEGMENT	62
REQUEST_CHOICE	67
REQUEST_LOCATOR	66
REQUEST_PICK	66
REQUEST_STRING	68
REQUEST_STROKE	67
REQUEST_VALUATOR	67
SAMPLE_CHOICE	69
SAMPLE_LOCATOR	68
SAMPLE_PICK	69
SAMPLE_STRING	69
SAMPLE_STROKE	68

SAMPLE_VALUATOR	68
SELECT_NORMALIZATION_TRANSFORMATION	61
SET_ASF	59
SET_CHAR_EXPANSION_FACTOR	58
SET_CHAR_HEIGHT	58
SET_CHAR_SPACING	58
SET_CHAR_UP_VECTOR	58
SET_CHOICE_MODE	66
SET_CLIPPING_INDICATOR	61
SET_COLOUR_REPRESENTATION	60
SET_DEFERRAL_STATE	53
SET_DETECTABILITY	63
SET_FILL_AREA_COLOUR_INDEX	59
SET_FILL_AREA_INDEX	58
SET_FILL_AREA_INTERIOR_STYLE	59
SET_FILL_AREA_REPRESENTATION	60
SET_FILL_AREA_STYLE_INDEX	59
SET_HIGHLIGHTING	63
SET_LINETYPE	57
SET_LINEWIDTH_SCALE_FACTOR	57
SET_LOCATOR_MODE	65
SET_MARKER_SIZE_SCALE_FACTOR	57
SET_MARKER_TYPE	57
SET_PATTERN_REFERENCE_POINT	59
SET_LINEWIDTH_SCALE_FACTOR	57
SET_LOCATOR_MODE	65
SET_MARKER_SIZE_SCALE_FACTOR	57
SET_MARKER_TYPE	57
SET_PATTERN_REFERENCE_POINT	59
SET_PATTERN_REPRESENTATION	60
SET_PATTERN_SJZE	59
SET_PICK_ID	59
SET_PICK_MODE	66
SET_POLYLINE_COLOUR_INDEX	57
SET_POLYLINE_INDEX	57
SET_POLYLINE_REPRESENTATION	59
SET_POLYMARKER_COLOUR_INDEX	57
SET_POLYMARKER_INDEX	57
SET_POLYMARKER_REPRESENTATION	60
SET_SEGMENT_PRIORITY	63
SET_SEGMENT_TRANSFORMATION	63
SET_STRING_MODE	66
SET_STROKE_MODE	65
SET_TEXT_ALIGNMENT	58
SET_TEXT_COLOUR_INDEX	58
SET_TEXT_FONT_AND_PRECISION	58
SET_TEXT_INDEX	57
SET_TEXT_PATH	58

**C. 178 TOCT P 34.1702.3-92**

SET_TEXT_REPRESENTATION	60
SET_VALUATOR_MODE	66
SET_VIEWPORT	61
SET_VIEWPORT_INPUT_PRIORITY	61
SET_VISIBILITY	63
SET_WINDOW	61
SET_WS_VIEWPORT	63
SET_WS_WINDOW	63
TEXT	55
UPDATE_WS	53
WRITE_ITEM_TO_GKSM	70

## ИНФОРМАЦИОННЫЕ ДАННЫЕ

1. ПОДГОТОВЛЕН И ВНЕСЕН ТК 22 «Информационная технология»
2. УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Постановлением Госстандарта России от 28.12.92. № 1577  
Настоящий стандарт подготовлен методом прямого применения международного стандарта ИСО 8651—3—88 «Системы обработки информации. Машинная графика. Связь ядра графической системы с языком программирования Ада»
3. ВВЕДЕН ВПЕРВЫЕ
4. ССЫЛОЧНЫЕ НОРМАТИВНО-ТЕХНИЧЕСКИЕ ДОКУМЕНТЫ

Обозначение НТД, на который дана ссылка	Номер пункта
ГОСТ 27017—88 ГОСТ 27831—88	Введение, 2, Приложение 2, Приложение

## СОДЕРЖАНИЕ

0. ВВЕДЕНИЕ	2
1. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ	2
2. ССЫЛКИ	3
3. СВЯЗЬ ЯДРА ГРАФИЧЕСКОЙ СИСТЕМЫ С ЯЗЫКОМ АДА	3
3.1. Условия соответствия стандарту	3
3.2. Включение в язык	4
3.2.1. Отображение функций	4
3.2.2. Реализация и зависимость от компьютера	4
3.2.3. Обработка ошибок	4
3.2.4. Отображение данных	5
3.2.5. Многозадачность	6
3.2.6. Пакирование	6
3.2.7. Среда прикладных программ	7
3.2.8. Регистрация	8
4. ТАБЛИЦЫ	8
4.1. Процедуры	8
4.2. Определение типов данных	25
4.2.1. Сокращения, используемые в определениях типов данных	25
4.2.2. Определение типов в алфавитном порядке	25
4.2.3. Список определений личных типов	46
4.2.4. Список деклараций констант	48
4.3. Коды ошибок	50
4.3.1. Задание кодов ошибок	51
4.3.2. Коды устраняемых ошибок	51
5. ФУНКЦИИ В АДЕ, СВЯЗАННЫЕ С ЯДРОМ ГРАФИЧЕСКОЙ СИСТЕМЫ	52
5.1. Функции ЯГС	52
5.2. Дополнительные функции	94
5.2.1. Подпрограммы для манипулирования записями входных данных	94
5.2.2. Пакет обобщенной координатной системы ЯГС	96
5.2.3. Общий пакет списка утилит ЯГС	98
5.2.4. Утилиты функций метафайла	101
5.3. Настраиваемые варианты	101
Приложение А. Спецификация скомпилированного ЯГС	102
Приложение Б. Список ссылок на определенные реализации записи	144
Приложение В. Примеры программ	145
Приложение Г. Многозадачный режим работы ЯГС	160
Приложение Д. Неподдерживаемые обобщенные примитивы вывода и расширения	165
Приложение Е. Типы элементов метафайла	168
Приложение Ж. Индексы функций ЯГС	170
Информационные данные	179

Редактор *Р. С. Федорова*  
Технический редактор *В. Н. Прусакова*  
Корректор *М. С. Кабанова*

Сдано в набор 22.01.93. Подп. в печ. 31.05.93. Усл. печ. л. 10,69 Усл. кр-отт. 10,82.  
Уч.-изд. л. 13,66. Тир. 341. С 231.

---

Ордена «Знак Почета» Издательство стандартов, 107076, Москва, Колодезный пер., 14.  
Калужская типография стандартов, ул. Московская, 256. Зак. 207

**к ГОСТ Р 34.1702.3—92 (ИСО 8651-3—88) Информационная технология. Машинная графика. Связь ядра графической системы с языком программирования Ада**

В каком месте	Напечатано	Должно быть
Обложка, титульный лист	ГОСТ 34.1702.3—92	ГОСТ Р 34.1702.3—92

(ИУС № 12 2007 г.)